

# MIXTURE OF PARROTS: EXPERTS IMPROVE MEMORIZATION MORE THAN REASONING

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

The Mixture-of-Experts (MoE) architecture enables a significant increase in the total number of model parameters with minimal computational overhead. However, it is not clear what performance tradeoffs, if any, exist between MoEs and standard dense transformers. In this paper, we show that as we increase the number of experts (while fixing the number of active parameters), the memorization performance consistently increases while the reasoning capabilities saturate.

We begin by analyzing the theoretical limitations of MoEs at reasoning. We prove that there exist graph problems that cannot be solved by any number of experts of a certain width; however, the same task can be easily solved by a dense model with a slightly larger width. On the other hand, we find that on memory-intensive tasks, MoEs can effectively leverage a small number of active parameters with a large number of experts to memorize the data. We empirically validate these findings on synthetic graph problems and memory-intensive closed book retrieval tasks. Lastly, we pre-train a series of MoEs and dense transformers and evaluate them on commonly used benchmarks in math and natural language. We find that increasing the number of experts helps solve knowledge-intensive tasks, but fails to yield the same benefits for reasoning tasks.

## 1 INTRODUCTION

The explosion in capabilities of large language models in recent years has largely been enabled by scaling their size, as measured by the number of parameters in the model. In the standard Transformer architecture, scaling the number of parameters entails a proportional increase in computational cost, e.g. doubling the number of parameters requires doubling the number of floating-point operations (FLOPs), making training and inference more computationally intensive. Mixture-of-Experts (MoE) were introduced as a solution for this problem (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022). MoEs replace the single MLP in each Transformer block with multiple MLPs (called experts), where each token is routed to a few experts based on a linear routing function. The number of parameters in the MoE layer therefore increases with the total number of experts, while the compute increases only with the number of “active” experts (i.e., the number of experts to which the token is routed to). This offers a promising option for scaling models: increase the number of experts instead of the model dimension or its depth. For this reason, MoEs have become very popular, and many frontier models today are based on the MoE architecture (Achiam et al., 2023; Databricks, 2023; Anil et al., 2023; Dai et al., 2024; Jiang et al., 2024; Yang et al., 2024).

In this work we study whether MoE indeed offers a “free-lunch”, enabling gains in performance with no computational cost. Interestingly, we find that the benefit from MoEs greatly depends on the task at hand. We show that for reasoning-based tasks, such as graph problems and mathematical reasoning, MoEs offer limited performance gains, and increasing the number of experts cannot compete with scaling the dimension (width) of the model. On the other hand, for memory-intensive tasks, we show that scaling the number of experts is competitive with scaling standard “dense” MLPs.

To demonstrate these claims, we begin with a theoretical analysis of MoEs and dense models. We use communication-complexity lower bounds to show that a single-layer MoE requires a critical dimension to solve a simple graph connectivity problem, implying that MoEs offer no benefit for solving this problem and only consume unnecessary memory. On the other hand, we show that for a pure memorization task, where the model only needs to “remember” an arbitrary set of examples,

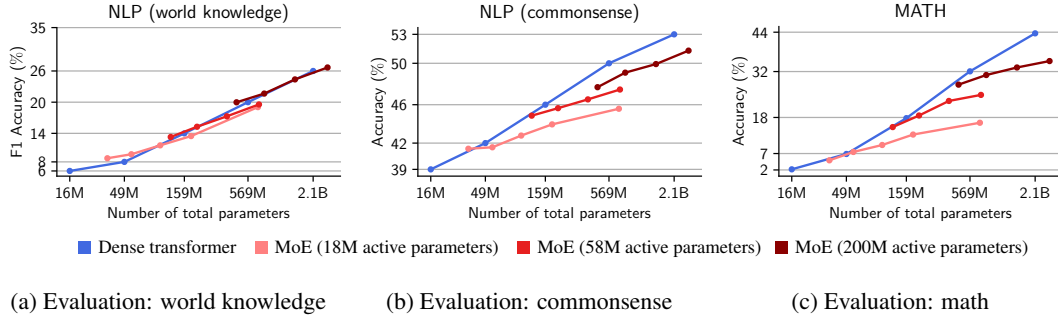


Figure 1: **(a) Evaluation: world knowledge.** We train a series of dense transformers and MoEs on 65B tokens from a corpus essentially made of Fineweb-edu, Cosmopedia and Wikipedia (see Section 5 for details). We then evaluate the models on several world knowledge benchmarks (e.g., TriviaQA (Joshi et al., 2017), Natural Questions (Kwiatkowski et al., 2019)) and report the average F1 accuracy. Surprisingly, at a fixed number of total parameters, MoEs with substantially fewer active parameters approximately match the performance of dense models. This highlights the importance of experts in tasks that require memorization. **(b) Evaluation: commonsense.** Here we evaluate the aforementioned pre-trained models on natural language commonsense benchmarks (e.g., HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021)). On these reasoning tasks, we observe that MoEs perform worse than dense models and more significant benefits are obtained by increasing the number of active parameters. **(c) Evaluation: math.** Here we train a series of dense transformers and MoEs on 65B tokens from a corpus essentially made of Proof-Pile2 (Azerbayev et al., 2023) (see Section 5 for details). The results are consistent with the ones in (b): MoEs perform worse than dense models at equal number of total parameters.

scaling the number of experts is equivalent to scaling the number of parameters in dense transformers, implying a significant computational gain when fixing the number of active parameters (Section 3). We continue by experimentally validating these results, comparing MoEs against dense models on synthetic training data. We train these models on finding the shortest path in random graphs, where we show that MoE accuracy does not improve as we increase the number of experts, but that accuracy consistently increases with width for a dense transformer (Figure 4b). Following this, we train different models on the task of memorizing a large phone-book. We demonstrate that MoEs excel in memorization, matching the performance of dense transformers with the same number of total parameters but with substantially less computational cost (Figure 4a).

Finally, we train dense transformers and MoEs on real datasets of mathematical reasoning and natural language, and perform intensive benchmarking of these models on a wide variety of downstream tasks. For memory-intensive tasks, MoEs surprisingly have a great advantage, where increasing the number of experts can match the performance of large dense models (Figure 1a). However, we show that for tasks that rely on reasoning, scaling the number of experts cannot compete with increasing the model dimension (Figures 1b-1c). Moreover, MoEs exhibit some memorization behaviors when trained on math problems (Figure 5). Taken together, our results show that the gains from using MoEs depend greatly on the nature of the training data and downstream task, and that while MoEs can improve performance in certain cases, sometimes increasing the effective size (width) of the model is unavoidable.

## 2 RELATED WORK

**Mixture of Experts.** Mixture-of-Experts (MoE) date back to the work of Jacobs et al. (1991); Jordan & Jacobs (1994). Shazeer et al. (2017); Fedus et al. (2022) were the first to scale this idea to deep learning and obtain state-of-the-art models in machine translation. Since then, several works have improved their routing algorithms (Lepikhin et al., 2020; Lewis et al., 2021; Roller et al., 2021; Clark et al., 2022; Zhou et al., 2022; Antoniak et al., 2023; Zhong et al., 2024), have improved their downstream performance after finetuning (Du et al., 2022; Zoph et al., 2022) or made their training and inference more efficient (Rajbhandari et al., 2022; Gale et al., 2023; Pan et al., 2024; Tan et al., 2024). However, only a few papers have studied the science of MoEs and their comparison with

dense transformers. Clark et al. (2022); Krajewski et al. (2024) establish scaling laws for MoEs. Chen et al. (2022) design a specific classification problem where a model with multiple experts provably outperforms one with only one expert. Shazeer et al. (2017); Lepikhin et al. (2020); Artetxe et al. (2021); Lewis et al. (2021); Fedus et al. (2022); Du et al. (2022) show that given a fixed FLOP budget, MoEs are always better. However, these papers claim that on a per parameter basis, MoEs always seem comparatively worse than dense models. In this paper, we temper this claim by showing that it depends on the *nature of the task* at hand: on reasoning tasks, we validate this claim but on memory-intensive tasks, equally-sized MoEs perform as well as dense transformers.

**Language models and memorization.** Large language models (LLMs) store a considerable amount of knowledge in their parameters (Petroni et al., 2019; Heinzerling & Inui, 2020). They memorize useful knowledge such as facts and commonsense (Zhao et al., 2024). Many works studied how memorization occurs in LLMs by developing tools to locate the knowledge in the model (Meng et al., 2022; Allen-Zhu & Li, 2023; Liu et al., 2024) or by tracking the training dynamics (Tirumala et al., 2022; Speicher et al., 2024). We draw inspiration from Allen-Zhu & Li (2023) and evaluate the memorization of our models by pre-training them on a mixture of datasets that includes Wikipedia, and at test time, evaluate them on world knowledge benchmarks, which are essentially question answering tasks on Wikipedia facts. With respect to theoretical findings, Kim et al. (2023); Mahdavi et al. (2023); Madden et al. (2024) provide upper bounds on the number of parameters needed for dense transformers to perform memorization tasks under various conditions.

**Language models and reasoning.** In recent years, transformer-based language models have displayed remarkable effectiveness in solving a broad range of reasoning tasks. Specifically, the reasoning capabilities of transformers have been studied in the context of arithmetic problems (Jelassi et al., 2023; Cho et al., 2024; Hou et al., 2024; Zhou et al., 2024; McLeish et al., 2024; Lee et al., 2023), mathematical reasoning (Zhang et al., 2022; Imani et al., 2023; Wei et al., 2022) graph problems (Sanford et al., 2024; Fatemi et al., 2023; Jin et al., 2023; Wang et al., 2024) and code challenges (Shi et al., 2024; Zhu et al., 2024). Recently, state-of-the-art language models were used for solving complex math olympiad problems (DeepMind, 2024; NuminaMath, 2024; OpenAI, 2024). With respect to theoretical findings, various works study the reasoning capabilities of transformers, relating their expressive power to other complexity classes and formal languages (Weiss et al., 2021; Zhou et al., 2023; Strobl et al., 2024). Other works study how chain-of-thought can improve the reasoning capabilities of language models in terms of expressive power and learnability (Abbe et al., 2024; Merrill & Sabharwal, 2023; Malach, 2023). However, the reasoning capabilities of MoE language models compared to their dense counterparts have received comparatively less attention.

### 3 THEORY: REPRESENTATIONAL CAPACITY

In this section, we analyze the capability of MoE transformers compared to standard (dense) models. We begin by studying a simple graph problem that requires scaling the hidden dimension of the transformer, showing that MoEs with small hidden dimension cannot solve this problem, regardless of the number of experts used. Then, we show that MoEs can effectively memorize random inputs, requiring significantly less computational resources (active parameters) compared to dense models.

#### 3.1 SETTING

Consider a one-layer transformer  $f \in \text{Transformer}_{m,H,1}^N$  which takes as input a sequence of length  $N$  and has logarithmic bit-precision.  $f$  embeds the input into dimension  $m$  via the function  $\phi$ .  $f$  has  $H \geq 1$  attention heads, whose outputs are combined via concatenation before we apply point-wise function  $\psi$ <sup>1</sup>.  $f$  is a *dense* transformer, if  $\psi$  is an MLP, i.e. function of the form:

$$\psi(\mathbf{x}) = \mathbf{u}^\top \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}), \text{ for } \mathbf{W} \in \mathbb{R}^{m' \times m}, \mathbf{b} \in \mathbb{R}^{m'}, \mathbf{u} \in \mathbb{R}^{m'}$$

<sup>1</sup>In multi-layer Transformers, each layer outputs a vector of size  $m$ . However, since our focus in this section will be on binary classification problems, we will let the transformer output a single scalar, and we interpret the output of the final token as the prediction for the classification task.

where  $\sigma$  is the ReLU activation function.  $f \in \text{Transformer}_{m,H,1,K}^N$  is an MoE transformer with  $K$  experts if  $\psi$  is a function of the form:

$$\psi(\mathbf{x}) = \mathbf{u}_i^\top \sigma(\mathbf{W}_i \mathbf{x} + \mathbf{b}_i) \text{ for } i = \underset{j}{\operatorname{argmax}} \mathbf{r}_j^\top \mathbf{x}$$

where  $\mathbf{W}_1, \dots, \mathbf{W}_k \in \mathbb{R}^{m' \times m}$ ,  $\mathbf{b}_1, \dots, \mathbf{b}_k \in \mathbb{R}^{m'}$ ,  $\mathbf{u}_1, \dots, \mathbf{u}_k \in \mathbb{R}^{m'}$  are the parameters of each expert and  $r_1, \dots, r_k$  define the routing function (we use top-1 routing).

Define the parameters as  $Q_h, V_h, K_h \in \mathbb{R}^{m \times m}$ ,  $\phi : \mathcal{X} \rightarrow \mathbb{R}^m$ ,  $\psi : \mathbb{R}^m \rightarrow \mathbb{R}$ . The output of  $f$  is:

$$f(\mathbf{x}_1, \dots, \mathbf{x}_N) = \psi \left( \left[ \operatorname{softmax}(\phi(\mathbf{x}_N)^\top Q_h K_h^\top \phi(X)) \phi(X) V_h \right]_{h \in [H]} \right).$$

### 3.2 MOES REQUIRE A CRITICAL HIDDEN SIZE TO SOLVE GRAPH REASONING TASKS

In this section, we analyze the graph reasoning capabilities of dense and sparse transformers. We define the length-2 path problem on a graph, and use it as a means to understand other graph reasoning tasks such as graph connectivity, shortest path, and cycle detection.

**Definition 3.1** (Length-2 Path Problem). The inputs is a graph  $G = (V, E)$ . The source  $s \in V$  and a destination  $d \in V$  are fixed for all tasks as the 0 and  $|V|$  vertex. The length-2 path problem asks whether there is a path of length 2 from  $s$  to  $d$ .

Graph connectivity, shortest path, and cycle detection are all graph reasoning tasks which reduce to the length-2 path problem due to (Sanford et al., 2024) and Lemma C.2. We provide a lower-bound on the width required for a sparse transformer to solve the length-2 path problem, and an upper-bound on the width required for a dense transformer to solve the problem. Further, we show a separation between dense and sparse transformers with the same number of parameters: for a sufficiently large amount of experts in the sparse model, it cannot solve the same problem that a dense model can solve with the same amount of *total* parameters.

**Lower bound on width of depth-1 MoE for reasoning.** We begin by showing a lower-bound on the width for a depth-1 mixture of expert model for the length-2 path problem. This lower bound implies a lower bound for search and retrieval tasks such as graph connectivity, shortest path, and cycle detection.

**Theorem 3.2** (Length-2 path lower-bound on sparse transformers). *For some input sequence  $G = (V, E)$ , fix two disjoint subsets  $A, B \subset [N - 1]$ , and consider a single-layer transformer  $f \in \text{Transformer}_{m,H,1,K}^N$  with  $O(\log N)$ -bit precision that solves length-2 path for any input  $X$  where  $X_A$  is a function of edges with the source  $s$ ,  $X_B$  is a function of edges with the destination  $d$ . Then,  $f$  has width satisfying  $mH = \Omega(|V|/\log N)$ .*

The proof follows almost identically from the proof in (Sanford et al., 2024) for the class  $\text{Transformer}_{m,H,1}^N$ . The original proof does not place constraints on the function  $\psi$  and is based on a communication-complexity argument. As such we may design  $\psi$  so that it first routes and then chooses which expert to apply. We give a complete proof in Appendix C. As such, the result of (Sanford et al., 2024) can also be extended to the class  $\text{Transformer}_{m,H,1,K}^N$ .

**Upper bound on width of depth-1 dense transformer for reasoning.** In this section we give an upper bound for the width required for a dense model to solve the length-2 path problem.

**Theorem 3.3** (Length-2 path width upper bound for transformer). *There exists a transformer of width  $|V|$  and  $O(\log N)$ -bit precision that solves length-2 path problem for any input.*

The proof relies on an encoding of the inputs where the output values only exceed a certain threshold when  $u$  and  $v$ , the source and destination vertices, have edges with a common vertex. We defer the proof to Appendix C.

**Parameter-matched comparison of dense and sparse depth-1 transformers.** Using the lower-bound on width required for a sparse transformer (Theorem 3.2) and the upper-bound on width required for a dense transformer (Theorem 3.3), we compare dense and sparse transformers when they have the same number of total parameters. We find that when the number of experts exceeds  $(\log N)^2$ , the sparse model is unable to solve the same task as the dense model.

**Corollary 3.4.** *Consider a sparse transformer (with  $K$  experts) and a dense transformer with the same number of parameters. There exists a number of experts  $K$  so that the sparse model is not able to solve the reasoning task, but the dense transformer solves the task.*

*Proof.* Suppose we have two depth-1 transformers, where one is a dense model and the other is a mixture of experts with  $K$  experts. Let the width of the dense model be  $m_d$ , and the width of the sparse model be  $m_s$ . The number of parameters in the dense model is  $O(m_d^2)$  and the number of parameters in the sparse model is  $O(Km_s^2)$ . In order to match the number of parameters, it must be the case that  $m_s = \frac{m_d}{\sqrt{K}}$ . Suppose we let  $m_d = |V|$ , as this is sufficient to solve the above problems. For any  $K \geq \Omega((\log N)^2)$ , the sparse model is not sufficiently wide to solve the problem.  $\square$

### 3.3 MOES USE THEIR EXPERTS TO SOLVE MEMORY-INTENSIVE TASKS

In this section, we provide an upper-bound on the number of parameters necessary for a sparse transformer to solve memorization tasks, followed by a lower-bound on the number of parameters needed for a dense transformer to solve the same task. We use these results to compare the memorization capabilities of dense and sparse transformers with the same number of active parameters. We find that with enough experts, the sparse transformer is able to solve memorization tasks with less active parameters than the dense transformer. In both bounds we assume that transformer has logarithmic number of bits to encode each parameter.

We consider sequences  $\{(X^i, y_i)\}_{i=1}^n$  where  $X^i \in \mathbb{R}^{N \times m}$  are input sequences of length  $N$  in dimension  $m$  such that  $X^i[j]$  is sampled from a Gaussian distribution  $\mathcal{N}(0, I_m)$ . We assume  $y_1, \dots, y_N \in \{\pm 1\}$  are arbitrary labels for the  $n$  sequences. The objective is for a transformer to memorize these sequences, i.e. map each input  $X^i$  to a label  $y_i$ . The classification is determined by the sign of the last token output.

**Upper-bound on MoE for memorization.** We begin by showing that, with high probability over the choice of the inputs, the MoE architecture can memorize (i.e., arbitrarily label the examples), with a small number of active parameters.

**Theorem 3.5.** *With probability at least 0.99, there exists a one-layer MoE transformer with  $K$  experts, using  $O\left(\frac{mn}{K} + mK\right)$  active parameters and  $O(mn + mK)$  total parameters that, when applied to each sequence  $X^i$ , outputs at the last token a value whose sign matches  $y_i$ , i.e.,  $\text{sign}(f(X_i)) = y_i$  for all  $i = 1, \dots, n$ .*

Specifically, if we choose  $K = \sqrt{n}$  we get that an MoE architecture can solve the memorization problem with  $O(m\sqrt{n})$  active parameters<sup>2</sup>. To prove this result, we show that for a random linear routing function, the number of examples routed to each expert is approximately  $n/K$ . Then, we show that an expert with  $O(n/K)$  neurons can memorize a sample of size  $O(n/K)$ . We present the full proof in Appendix C.

**Lower bound on memorization with dense Transformer.** Next, we give a lower-bound on the number of parameters for a dense transformer to perform memorization.

**Theorem 3.6** (Lower bound for dense model). *Given the same task as above, a dense Transformer requires  $\tilde{\Omega}(n)$  parameters to solve the memorization task.*

This bound follows from the fact that there are  $2^n$  possible labels for any fixed set of  $n$  inputs, and at most  $2^{cW}$  functions with  $W$  parameters and  $c$  bit per parameters.

**Separation between MoEs and Dense Models.** Observe that the previous results on memorization imply a separation between MoEs and dense models in terms of the number of active parameters. Namely, we showed that an MoE with  $O(m\sqrt{n})$  active parameters can memorize, while a dense model requires  $\tilde{\Omega}(n)$  parameters. So, for large enough  $n$  (i.e. when  $n \gg m^2$ ), MoEs are significantly more efficient. Comparing the number of total parameters, MoEs require  $O(mn)$  parameters (assuming  $K \leq n$ ), so both MoE and dense models have linear dependence on  $n$  in the total parameter count.

<sup>2</sup>We believe that this bound can be improved using more complicated analysis, to show that only  $O(\sqrt{n})$  parameters are required. However, we leave this result to future work.



## 4 SYNTHETIC EXPERIMENTS

In the previous section, we proved that there exist graph connectivity problems that cannot be solved by any number of experts of a certain width but the same task can be solved by a dense model with a slightly larger width. Our goal in this section is to verify that our theoretical analysis bears out experimentally when training models from scratch on synthetic data, before moving on to study pre-trained models in Section 5. We mainly focus on two tasks: the *shortest path* problem (Figure 2), which we use as a synthetic task to represent reasoning problems, and the *phone-book* task (Figure 3), to measure the memorization ability of our models. Our experiments in this section highlight that adding experts yields greater performance improvements on memorization tasks than reasoning tasks.

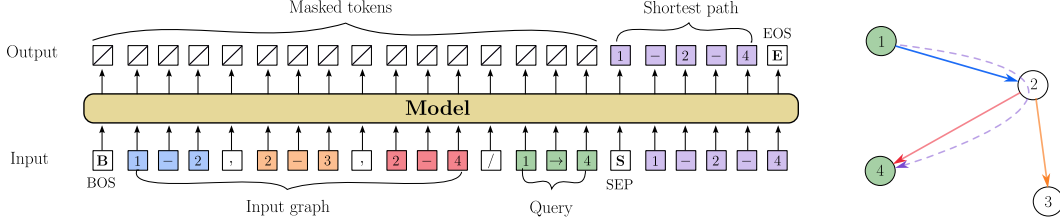


Figure 2: Illustration of the shortest path task. We feed the model with a sequence that lists all the edges in the input graph and ends with the query (in green) which asks the model to find a shortest path between two vertices (from vertex 1 to vertex 4 in the figure). The model then autoregressively returns the shortest path (in purple).

### 4.1 EXPERIMENTAL SETUP

**Architecture.** We opt for the Mistral (Jiang et al., 2023) and Mixtral (Jiang et al., 2024) architectures as the backbones of our Transformer and MoE models, respectively. The two architectures are identical and only differ by the addition of a gating module and multiple experts in Mixtral. For both model types, we fix the number of layers to  $L = 12$ . For the dense transformers, we vary model size by sweeping the width  $d \in \{256, 512, 1024\}$ . For MoEs, we sweep over widths  $d \in \{256, 512\}$  and the number of experts  $E \in \{8, 16, 32, 64\}$ . To be consistent with our experiments in Section 5, we set the intermediate dimension in the FFN block to be equal to  $d$  (and not  $4d$ ). We use token-choice routing, do not apply any token dropping and each token is routed to the top-2 experts. Lastly, in both this section and Section 5, we report for each model the number of non-embedding parameters which we refer to as the total number of parameters.

**Shortest path task.** For a graph with  $n$  vertices, our token space  $\mathcal{V}$  is of size  $n + 6$  with tokens encoding the vertices and some special tokens:  $\mathcal{V} = \{1, \dots, n, \langle \text{EDGE} \rangle, \langle \text{BOS} \rangle, \langle \text{EOS} \rangle, \langle \text{PAD} \rangle, \langle \text{SEP} \rangle, / \}$  where  $\langle \text{BOS} \rangle$  is the beginning of sentence token,  $\langle \text{EOS} \rangle$  the end of sentence token,  $\langle \text{PAD} \rangle$  the padding token,  $\langle \text{EDGE} \rangle$  is the token indicating an edge between two vertices and,  $\langle \text{SEP} \rangle$  and  $/$  are separator tokens. Each sequences describes the graph by a list of all the edges followed by two randomly sampled vertices and the shortest path between these latter (see Figure 2). All the graphs are directed and sampled according to the Erdős-Rényi model, with  $n$  vertices and probability  $p$  for each edge to exist. We vary  $n \in \{25, 30, 50, 40, 45, 50, 55\}$  and set  $p$  such that the average length of the shortest path is 3.5. Each train/test pair corresponds to *one* value of  $(n, p)$ , we do not mix graph sizes.

**Phone-book task.** Our token space  $\mathcal{V}$  is of size 39 and made of the alphabet letters, digits and special tokens:  $\mathcal{V} = \{a, \dots, z, 0, \dots, 9, \langle \text{BOS} \rangle, \langle \text{EOS} \rangle, \langle \text{SEP} \rangle\}$ . We generate phone-books where the names consist of 5 letters and the phone numbers of 8 digits (see Figure 3). We ensure that both the names and numbers are unique.

**Datasets.** For the graph experiments, the training set size is  $1e6$  and the test set consists of  $1e3$  held-out examples that are sampled from the *same* distribution as the training examples. For the phone-book experiments, we vary the training set size over  $\{1e5, 5e5, 1e6, 1.5e6, 2e6, 2.5e6, 3e6\}$  and the test set consists of  $1e3$  queries from the training set.

**Optimization.** We use the AdamW optimizer (Loshchilov et al., 2017) with a weight decay equal to 0.1. We sweep the learning rate over  $\{5e-5, 1e-4, 5e-4, 1e-3\}$ , the number of epochs over

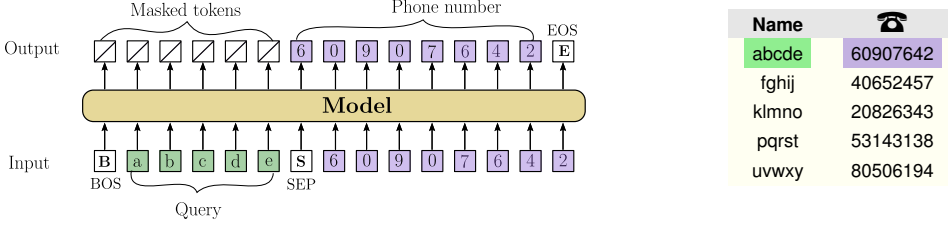


Figure 3: Illustration of the phone-book task for closed-book retrieval. The model is first trained to memorize a phone-book (illustrated on the right). Then, we randomly select a name in the phone-book (in green) and ask the model to return their phone number (in purple) without access to the phone-book.

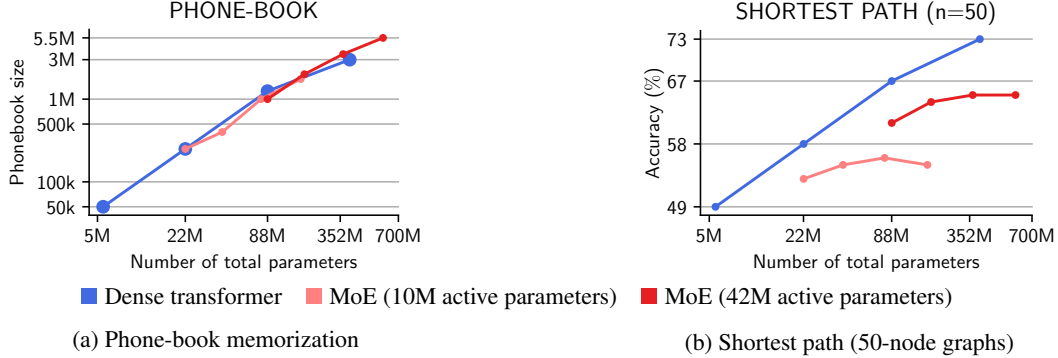


Figure 4: (a) **Phone-book memorization**: We train a series of dense transformers and MoEs on phone-books of varying sizes and then evaluate their memorization capacity. We report the maximal phone-book size where the model obtains more than 90% accuracy. The maximal phone-book size correlates with the total (and not active) number of parameters. (b) **Shortest path (total parameters)**: We train models to find the shortest path in 50-node graphs and report the test accuracy. Here, increasing the number of experts provides limited improvements and the performance rather correlates with the number of active parameters.

{2, 5, 10, 15}, and set the maximal possible batch size among {8, 16, 32}. We use a warmup during the 20% first training steps and a linear decay scheduler. All models are trained by next-token prediction. In the graph task, we apply a mask on the input instance so that we only penalize the model whenever it makes a mistake on the labels (and not on the inputs and labels jointly). In the phone-book experiment, we do not apply any masking.

**Evaluation.** For each task we compute the exact accuracy, i.e. we count the generation as correct only if it fully matches the ground truth. For the phone-book task, we report the size of the maximal phone-book where we observe at least 90% exact accuracy.

#### 4.2 MEMORIZATION: TOTAL PARAMETERS PREDICT PERFORMANCE

We train dense transformers and MoEs on phone-books of different sizes and at test time, evaluate whether they memorize the phone number of some names. Figure 4a reports the maximal phone-book size where the model manages to get an accuracy greater than 90%. This gives us an estimate of the memorization capacity of the model.

The findings are clear: no matter the number of active parameters, MoEs match the performance of dense transformers with the same number of total parameters. This suggests that MoEs are able to effectively leverage the extra parameters in additional experts by routing tokens to the experts that contain the necessary information from the training corpus. This scaling is remarkable in this case since it even holds when we are only routing to 2 out of 64 experts. For instance, we find that an MoE model with only 42M active parameters outperforms a dense model with 10x as many parameters. This type of impressively efficient memorization capacity may be a major reason behind the success of MoE architectures.

### 4.3 REASONING: TOTAL PARAMETERS DO NOT PREDICT PERFORMANCE

We train dense transformers and MoEs on the shortest path task and then query the models to find the shortest paths in novel, held-out graphs. Figure 4b reports the performance on graphs with 50 nodes with respect to their number of total parameters. Contrary to the phone-book experiment, increasing the number of experts does not consistently improve the performance of MoEs. Essentially, we find that active parameters rather than total parameters is a better predictor of performance for these reasoning tasks.

To connect back to the theory from Section 3, note that active parameters is directly determined by the width of the network since we always route to exactly 2 experts and fix the depth. Thus, these results corroborate the theory by showing that width (i.e. active parameters) determines the performance on these graph reasoning problems and that increasing the number of experts is not helpful. In Section 5, we will further corroborate this idea through evaluation of pre-trained models on commonsense and math reasoning benchmarks.

## 5 PRE-TRAINED MODELS

In this section, we pre-train dense transformers and MoEs and compare their performance on standard math and natural language benchmarks. We break the downstream tasks into those that require more memorization and those that require more reasoning. The memorization-intensive tasks test for “world knowledge” and consist of benchmarks like TriviaQA (Joshi et al., 2017). We break the reasoning-intensive tasks into two subcategories: one for natural language reasoning tasks like WinoGrande (Sakaguchi et al., 2021) and another for mathematical reasoning tasks like Hendrycks-MATH (Hendrycks et al., 2021). These tasks may be seen as real-world analogs of the stylized phone-book and shortest path tasks studied in Section 4.

We observe that performance on world-knowledge tasks is governed by the total number of parameters while performance on reasoning tasks depends more on the number of active parameters (Figure 1). Additionally, we conduct an experiment that indicates memorization from MoEs may be harming reasoning performance since there is a larger gap between train and test accuracy for MoEs than dense models at fixed total parameters (Figure 5). Finally, we conduct an ablation where we compare models at fixed validation perplexity rather than model size. We find that MoEs perform better on world knowledge tasks and similarly on reasoning tasks compared to dense models (Figure 6).

### 5.1 SETUP

**Architecture.** We train dense transformers and MoEs using the OLMoE codebase (Muenighoff et al., 2024). We set the number of layers  $L = 20$  and vary the width  $d \in \{256, 512, 1024, 2048, 4096\}$  for dense transformers and  $d \in \{256, 512, 1024\}$ . Similarly to Muenighoff et al. (2024), we consistently set the intermediate dimension in the FFN/MoE blocks to be equal to  $d$  (and not  $4d$ ). For MoEs, we vary the number of experts  $E \in \{8, 16, 32, 64\}$ . For the specific case of width 256, we also train a MoE with 256 experts because its parameter count approximately matches the one of a width-2048 dense model and thus, we can compare the downstream performance of the two models. We use top-2 token-choice routing, without token dropping which is implemented in the dMoE function from the Megablocks package (Gale et al., 2023).

**Training hyperparameters.** We use the AdamW optimizer (Loshchilov et al., 2017) with a weight decay equal to 0.1. We set the learning rate to 0.001, train on 63B tokens (60k steps) with batch size 512 and sequence length of 2048. We use warmup during the 20% first training steps and a linear decay scheduler. We train our models using FSDP (Zhao et al., 2023).

**Pre-training datasets.** We train two collections of models, one series on natural language and another one on math. The “natural language” dataset is a mixture constituted of FineWeb-edu (Penedo et al., 2024), Cosmopedia (Ben Allal et al., 2024), Wikipedia and the training sets of the downstream tasks we evaluate on. The “math” dataset is a mixture made of Proof-Pile 2 (Azerbayev et al., 2023) and instruction datasets such as OpenMathInstruct (Toshniwal et al., 2024) and MetaMathQA (Yu et al., 2023). Each of the two training mixture approximately totals 65B tokens. A precise description of the training mixtures can be found in Appendix A.



**Evaluation.** We measure the validation perplexity on 5,000 held-out sequences sampled from the training distribution. And we evaluate our models on a series of natural language and math benchmarks. Explicitly, we divide them into three categories:

- World-knowledge tasks: TriviaQA (Joshi et al., 2017), Natural Questions (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), WebQuestions (Berant et al., 2013), ComplexWebQuestions (Talmor & Berant, 2018).
- Commonsense tasks: ARC-C and ARC-E (Clark et al., 2018), CommonsenseQA (Talmor et al., 2018), HellaSwag (Zellers et al., 2019), OpenbookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), SIQA (Sap et al., 2019), WinoGrande (Sakaguchi et al., 2021).
- Math benchmarks: SVAMP (Patel et al., 2021), GSM8k (Cobbe et al., 2021), GSM-Hard (Gao et al., 2023), Hendrycks-MATH (Hendrycks et al., 2021) and Minerva-MATH (Lewkowycz et al., 2022).

In all our experiments, we plot the average accuracy for each of these three categories. We report the corresponding per-task performance in Appendix B.

## 5.2 RESULTS

### Experts improve memorization more than reasoning.

We observe that our theoretical results from Section 3 and the synthetic experiments also hold when pre-training and evaluating language models on natural language and math. In Figure 1a, we report the accuracy of our models with respect to the number of *total* parameters. All the lines in the plot approximately coincide which implies that regardless of the number of active parameters, MoEs can effectively use their routing to leverage all of their parameters to solve memory-intensive tasks. On the other hand, on commonsense and math benchmarks (Figures 1b, 1c) we find that MoEs do not reach the performance of dense models with the same number of total parameters. This indicates that for these reasoning tasks, increasing the dense model width is more effective than adding experts.

### On mathematics tasks, MoEs display a higher train-test gap than dense models, suggestive of memorization.

We provide additional evidence that memorization occurs in pre-trained MoEs by considering the generalization gap. In Figure 5 we select 6,319 random problems from the OpenMathInstruct dataset, which is part of the training mixture data. More precisely, we pick 5,000 Hendrycks-MATH like examples and 1,319 GSM8k-like examples to ensure that the number of training examples matches with the corresponding number of examples in GSM8k and Hendrycks-MATH test sets. We then report the *generalization gap*, which is the gap between the accuracy on training examples and test examples. Despite making a *single* pass on the OpenMathInstruct dataset, Figure 5 shows that at scales beyond 159M parameters, MoEs suffer from a more significant generalization gap than dense transformers. This is suggestive that MoEs are more liable to memorize training data than dense models.

### MoE models excel at world knowledge tasks but match dense models in reasoning when perplexity is fixed.

Finally, we focus on the relationship between validation perplexity and downstream performance in Figure 6. Rather than comparing models by their parameter count, we can compare them based on how well they fit the training distribution as measured by validation perplexity. Even though two models may have the same perplexity, they will have learned different functions. The question is then if we can see any high level patterns in which types of functions a particular model class is more likely to learn. Figure 6a shows that at a fixed perplexity, the MoE models outperform the dense models on world knowledge tasks. This suggests that MoEs do have a bias towards learning functions that memorize training data. On the other hand, Figures 6b and 6c show that MoEs and

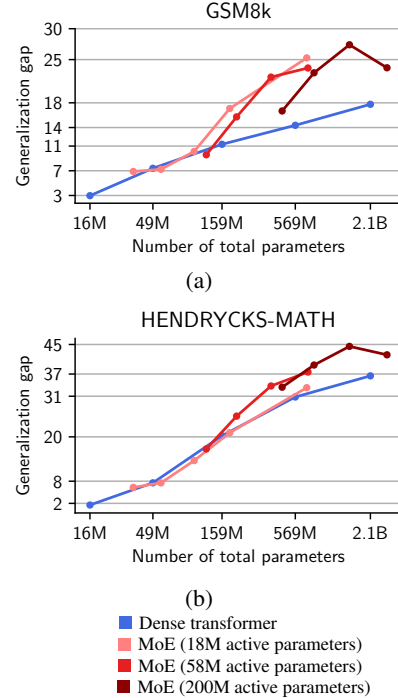


Figure 5: Generalization gap i.e., difference between the training and test accuracies, when the test set is GSM8k (a) and Hendrycks-MATH (b).

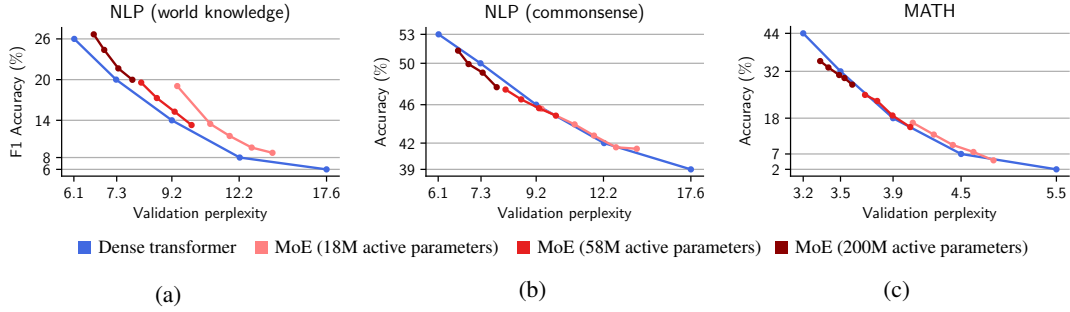


Figure 6: (a) On world knowledge benchmarks, MoEs consistently outperform dense transformers in downstream performance when fixing the validation perplexity. (b-c) In reasoning benchmarks, dense transformers perform about the same as MoEs at a fixed validation perplexity. MoEs can achieve these perplexities with less active parameters, but may require substantially more total parameters.

dense models perform about the same on the reasoning tasks at fixed validation perplexity. We can square this with the results from Figure 1 by noting that at equal total number of parameters an MoE has worse validation perplexity than the corresponding dense model. This suggests that while MoEs do not change the relationship between perplexity and downstream accuracy on reasoning tasks relative to dense models, they may struggle to learn the reasoning parts of the training distribution as well.

Overall, our main findings in Figure 1 and supplementary experiments in Figures 5 and 6 corroborate the hypothesis that MoEs can effectively use more experts to increase their memory capacity, but not necessarily their capability to reason.

## 6 DISCUSSION

In recent years, scaling up the number of parameters in Transformers has been the dominant approach for improving performance on language modeling. A standard Transformer of dimension  $d$  and sequence length  $L$  has number of parameters which scales with  $O(d^2)$ , and run-time that scales with  $O(d^2L^2)$ . Improving the efficiency can either attempt to reduce the dependence on  $L$  (Katharopoulos et al., 2020; Peng et al., 2023; Fu et al., 2022; Gu & Dao, 2023), while MoEs attempt to improve dependence on  $d$  by scaling the number of parameters without scaling the dimension of the model.

This paper illuminates the costs and benefits of this reduced dependence on  $d$ . We show that for some reasoning-intensive tasks increasing the dimension  $d$  is inevitable, and scaling the computation with  $O(d^2)$  seems unavoidable. This remains true regardless of the different design choices in the MoE architecture and is backed up empirically. We note that there is increasing interest in developing non-MoE models with sub-quadratic dependence on  $d$ , using some structural assumptions on the weight layers (Kamalakara et al., 2022; Dao et al., 2021; 2022; Fu et al., 2024), which could provide an alternative.

On the other hand, we find that MoEs are highly effective at knowledge intensive tasks. They are able to much more efficiently memorize facts than dense models with a similar number of active parameters, even matching the performance of dense models with the same number of total parameters. This suggests that MoEs are valuable as memorization machines and perhaps this particular capability can be leveraged while relying on other architectures for more reasoning-intensive tasks.

**Limitations and future work.** While we provide substantial experiments on pre-trained models with up to  $\leq 2.1\text{B}$  parameters, we recognize that large scale MoEs like Mixtral (Jiang et al., 2024), DeepSeek-V2 (Dai et al., 2024), and others have orders of magnitude more parameters. We hypothesize that our results would still be meaningful at larger scales due to the strong theoretical underpinning, but it is not guaranteed. Moreover, as suggested above, it would be an interesting direction for future work to propose new architectures with reduced  $d$  dependence that can get the best of both worlds and solve reasoning and memorization tasks.

## REFERENCES

- Emmanuel Abbe, Samy Bengio, Aryo Lotfi, Colin Sandon, and Omid Saremi. How far can transformers reason? the locality barrier and inductive scratchpad, 2024. URL <https://arxiv.org/abs/2406.06467>.
- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023.
- Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 1, 2023.
- Szymon Antoniak, Sebastian Jaszczur, Michał Krutul, Maciej Pióro, Jakub Krajewski, Jan Ludziejewski, Tomasz Odrzygóźdź, and Marek Cygan. Mixture of tokens: Efficient llms through cross-example aggregation. *arXiv preprint arXiv:2310.15961*, 2023.
- Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, et al. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684*, 2021.
- Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics, 2023.
- Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. Cosmopedia, 2024. URL <https://huggingface.co/datasets/HuggingFaceTB/cosmopedia>.
- Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1533–1544, 2013.
- Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.
- Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. Towards understanding mixture of experts in deep learning. *arXiv preprint arXiv:2208.02813*, 2022.
- Hanseul Cho, Jaeyoung Cha, Pranjal Awasthi, Srinadh Bhojanapalli, Anupam Gupta, and Chulhee Yun. Position coupling: Leveraging task structure for improved length generalization of transformers. *arXiv preprint arXiv:2405.20671*, 2024.
- Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. In *International conference on machine learning*, pp. 4057–4086. PMLR, 2022.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

- Tri Dao, Beidi Chen, Kaizhao Liang, Jiaming Yang, Zhao Song, Atri Rudra, and Christopher Re. Pixelated butterfly: Simple and efficient sparse training for neural network models. *arXiv preprint arXiv:2112.00029*, 2021.
- Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*, pp. 4690–4721. PMLR, 2022.
- Databricks. Introducing dbrx: A new state-of-the-art open llm. *Databricks Blog*, 2023. URL <https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm>. Accessed: 2023-10-12.
- DeepMind. Ai achieves silver-medal standard solving international mathematical olympiad problems. <https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/>, 2024.
- Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.
- Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560*, 2023.
- William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.
- Dan Fu, Simran Arora, Jessica Grogan, Isys Johnson, Evan Sabri Eyuboglu, Armin Thomas, Benjamin Spector, Michael Poli, Atri Rudra, and Christopher Ré. Monarch mixer: A simple sub-quadratic gemm-based architecture. *Advances in Neural Information Processing Systems*, 36, 2024.
- Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.
- Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5:288–304, 2023.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.
- Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.
- Benjamin Heinzerling and Kentaro Inui. Language models as knowledge bases: On entity representations, storage capacity, and paraphrased queries. *arXiv preprint arXiv:2008.09036*, 2020.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.
- Kaiying Hou, David Brandfonbrener, Sham Kakade, Samy Jelassi, and Eran Malach. Universal length generalization with turing programs. *arXiv preprint arXiv:2407.03310*, 2024.
- Shima Imani, Liang Du, and Harsh Shrivastava. Mathprompter: Mathematical reasoning using large language models. *arXiv preprint arXiv:2303.05398*, 2023.
- Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

- Samy Jelassi, Stéphane d’Ascoli, Carles Domingo-Enrich, Yuhuai Wu, Yanzhi Li, and Francois Charton. Length generalization in arithmetic transformers. *arXiv preprint arXiv:2306.15400*, 2023.
- Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
- Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.
- Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. Large language models on graphs: A comprehensive survey. *arXiv preprint arXiv:2312.02783*, 2023.
- Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.
- Siddhartha Rao Kamalakara, Acyr Locatelli, Bharat Venkitesh, Jimmy Ba, Yarin Gal, and Aidan N Gomez. Exploring low rank training of deep neural networks. *arXiv preprint arXiv:2209.13569*, 2022.
- Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Francois Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.
- Junghwan Kim, Michelle Kim, and Barzan Mozafari. Provable memorization capacity of transformers. In *The Eleventh International Conference on Learning Representations*, 2023.
- Jakub Krajewski, Jan Ludziejewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, et al. Scaling laws for fine-grained mixture of experts. *arXiv preprint arXiv:2402.07871*, 2024.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.
- Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*, 2023.
- Dmitry Lepikhin, HyukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.
- Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pp. 6265–6274. PMLR, 2021.
- Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.
- Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. Tinygsm: achieving 80% on gsm8k with small language models. *arXiv preprint arXiv:2312.09241*, 2023.
- Yan Liu, Yu Liu, Xiaokang Chen, Pin-Yu Chen, Daoguang Zan, Min-Yen Kan, and Tsung-Yi Ho. The devil is in the neurons: Interpreting and mitigating social biases in language models. In *The Twelfth International Conference on Learning Representations*, 2024.



- Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5, 2017.
- Liam Madden, Curtis Fox, and Christos Thrampoulidis. Upper and lower memory capacity bounds of transformers for next-token prediction. *arXiv preprint arXiv:2405.13718*, 2024.
- Sadegh Mahdavi, Renjie Liao, and Christos Thrampoulidis. Memorization capacity of multi-head attention in transformers. *arXiv preprint arXiv:2306.02010*, 2023.
- Eran Malach. Auto-regressive next-token predictors are universal learners. *arXiv preprint arXiv:2309.06979*, 2023.
- Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. Transformers can do arithmetic with the right embeddings, 2024. URL <https://arxiv.org/abs/2405.17399>.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.
- William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.
- Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math, 2024.
- Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. Olmoe: Open mixture-of-experts language models, 2024. URL <https://arxiv.org/abs/2409.02060>.
- NuminaMath. How numinamath won the 1st aimo progress prize. <https://huggingface.co/blog/winning-aimo-progress-prize>, 2024.
- OpenAI. Introducing openai o1. <https://openai.com/o1/>, 2024.
- Bowen Pan, Yikang Shen, Haokun Liu, Mayank Mishra, Gaoyuan Zhang, Aude Oliva, Colin Raffel, and Rameswar Panda. Dense training, sparse inference: Rethinking training of mixture-of-experts language models. *arXiv preprint arXiv:2404.05567*, 2024.
- Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open dataset of high-quality mathematical web text. *arXiv preprint arXiv:2310.06786*, 2023.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.
- Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024.
- Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.
- Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

- Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, pp. 18332–18346. PMLR, 2022.
- Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34:17555–17566, 2021.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.
- Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. Understanding transformer reasoning capabilities via graph algorithms. *arXiv preprint arXiv:2405.18512*, 2024.
- Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.
- David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.
- Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.
- Quan Shi, Michael Tang, Karthik Narasimhan, and Shunyu Yao. Can language models solve olympiad programming? *arXiv preprint arXiv:2404.10952*, 2024.
- Till Speicher, Aflah Mohammad Khan, Qinyuan Wu, Vedant Nanda, Soumi Das, Bishwamittra Ghosh, Krishna P Gummadi, and Evimaria Terzi. Understanding the mechanics and dynamics of memorisation in large language models: A case study with random strings. 2024.
- Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024.
- Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*, 2018.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.
- Shawn Tan, Yikang Shen, Rameswar Panda, and Aaron Courville. Scattered mixture-of-experts implementation. *arXiv preprint arXiv:2403.08245*, 2024.
- Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35:38274–38290, 2022.
- Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint arXiv:2402.10176*, 2024.
- Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36, 2024.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

- Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11080–11090. PMLR, 2021. URL <http://proceedings.mlr.press/v139/weiss21a.html>.
- Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*, 2017.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.
- Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.
- Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhui Chen. Mammoth2: Scaling instructions from the web. *arXiv preprint arXiv:2405.03548*, 2024.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.
- Yi Zhang, Arturs Backurs, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv:2206.04301*, 2022.
- Yifan Zhang. Stackmathqa: A curated collection of 2 million mathematical questions and answers sourced from stack exchange, 2024. URL <https://huggingface.co/datasets/math-ai/StackMathQA>.
- Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.
- Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36, 2024.
- Zexuan Zhong, Mengzhou Xia, Danqi Chen, and Mike Lewis. Lory: Fully differentiable mixture-of-experts for autoregressive language model pre-training. *arXiv preprint arXiv:2405.03133*, 2024.
- Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization, 2023. URL <https://arxiv.org/abs/2310.16028>.
- Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.
- Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. Transformers can achieve length generalization but not robustly. *arXiv preprint arXiv:2402.09371*, 2024.
- Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y Wu, Yukun Li, Huazuo Gao, Shirong Ma, et al. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931*, 2024.
- Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

## A DETAILS ON THE PRE-TRAINING DATASETS

In [Section 5](#), we pretrain two collections of models, one on “natural language” and the other on “math”. Here, we give a precise breakdown of our training mixtures. We start with the “natural language” training mixture that totals 64B tokens:

- 37B tokens from Fineweb-edu dedup ([Penedo et al., 2024](#)).
- 14B tokens from Cosmopedia ([Ben Allal et al., 2024](#)).
- 12B tokens from Wikipedia (we loop over Wikipedia 3 times).
- 1B tokens from the training set of the downstream tasks we test on. We create 3 copies of each of these to increase their presence in the mixture. The presence of these datasets is pretty important as argued in [Allen-Zhu & Li \(2023\)](#) so that the model is familiar with the downstream tasks at test time.
  - \* ComplexWebQuestions training set ([Talmor & Berant, 2018](#))
  - \* HotPotQA training set ([Yang et al., 2018](#))
  - \* Natural Questions training set ([Kwiatkowski et al., 2019](#))
  - \* TriviaQA training set ([Joshi et al., 2017](#))
  - \* WebQuestions training set ([Berant et al., 2013](#))
  - \* ARC-Easy and ARC-Challenge training sets ([Clark et al., 2018](#))
  - \* Hellaswag training set ([Zellers et al., 2019](#))
  - \* OpenBookQA training set ([Mihaylov et al., 2018](#))
  - \* PIQA training set ([Bisk et al., 2020](#))
  - \* SciQ training set ([Welbl et al., 2017](#))
  - \* SIQA training set ([Sap et al., 2019](#))
  - \* Winogrande training set ([Sakaguchi et al., 2021](#))

Our “math” training mixture that totals 66B tokens gathers:

- 55B tokens from Proof-Pile 2 ([Azerbaiyev et al., 2023](#)) that contain AlgebraicStack (11B), OpenWebMath ([Paster et al., 2023](#)) and ArXiv (29B).
- 2B tokens from OpenMathInstruct-1: we select the instances with a correct answer from the training set ([Toshniwal et al., 2024](#))
- 7B tokens from DeepMind math ([Saxton et al., 2019](#))
- 2B tokens from the following instruction-like datasets:
  - \* Math-Orca ([Mitra et al., 2024](#))
  - \* TinyGSM ([Liu et al., 2023](#)) (we only select 1 million examples from there).
  - \* StackMathQA ([Zhang, 2024](#))
  - \* MAmmoTH2 ([Yue et al., 2024](#)) (we only select the mathstackexchange subset).
  - \* NuminaMath-CoT ([NuminaMath, 2024](#)) (duplicated 3 times)
  - \* MetaMathQA ([Yu et al., 2023](#)) (duplicated 3 times)

## B ADDITIONAL EXPERIMENTS

In all our experiments in [Section 5](#), we report the average accuracy performance obtained by our pre-trained models on respectively world knowledge, commonsense and math benchmarks. Here, we provide the results per task. In [Subsection B.1](#), we display for each task, the downstream performance on a per parameter basis (similar to [Figure 1](#)) and in [Subsection B.2](#), we plot for each task, the downstream performance on a per validation perplexity basis (similar to [Figure 6](#)).

### B.1 DOWNSTREAM PERFORMANCE ON A PER PARAMETER BASIS

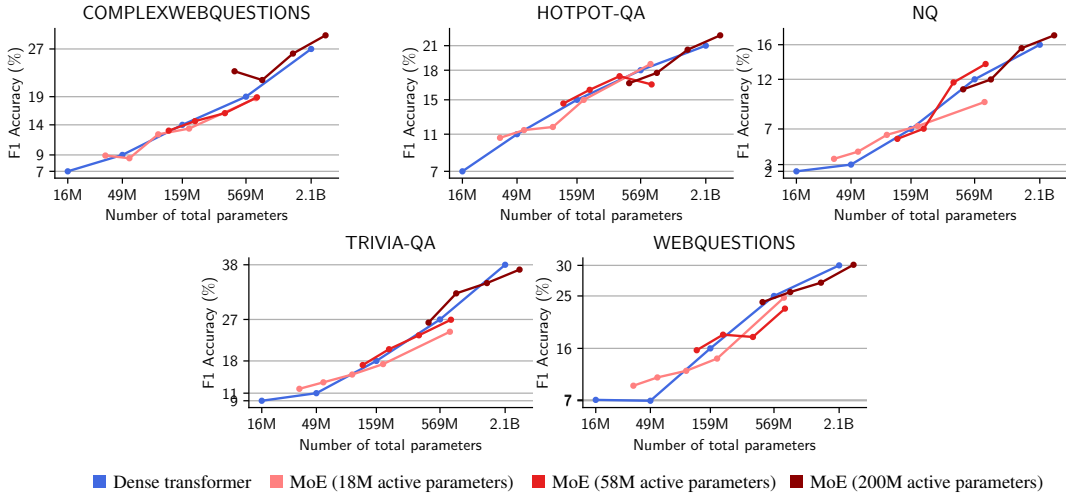


Figure 7: Downstream performance on the world knowledge tasks with respect to the total number of parameters of the models.

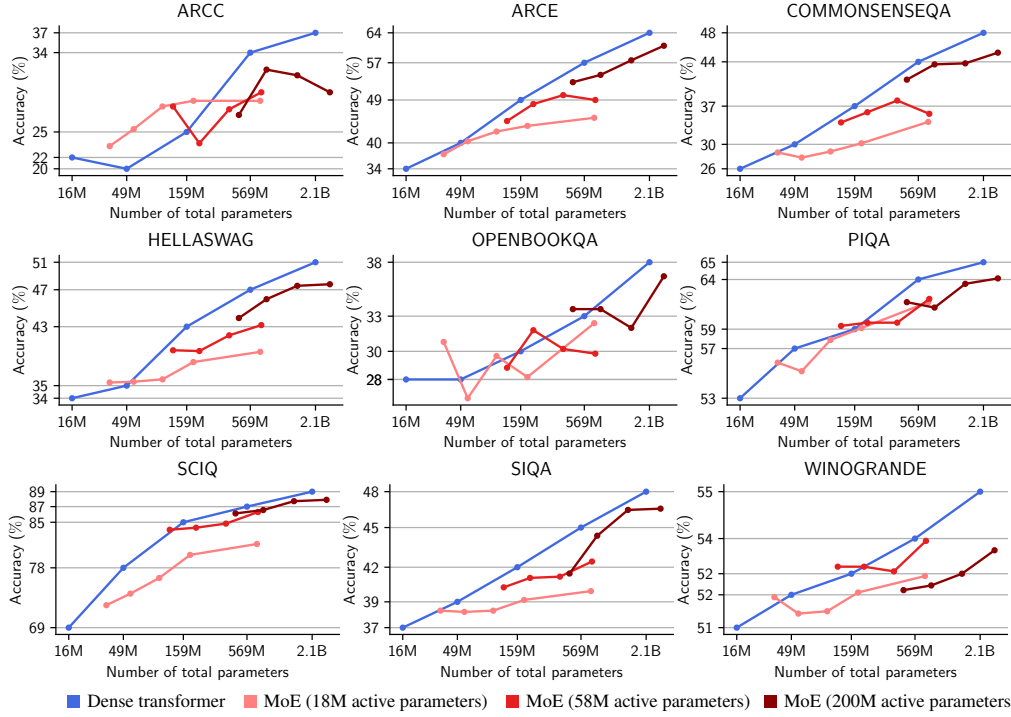


Figure 8: Downstream performance on the commonsense tasks with respect to the total number of parameters of the models.

## B.2 DOWNSTREAM PERFORMANCE ON A PER VAL PERPLEXITY BASIS



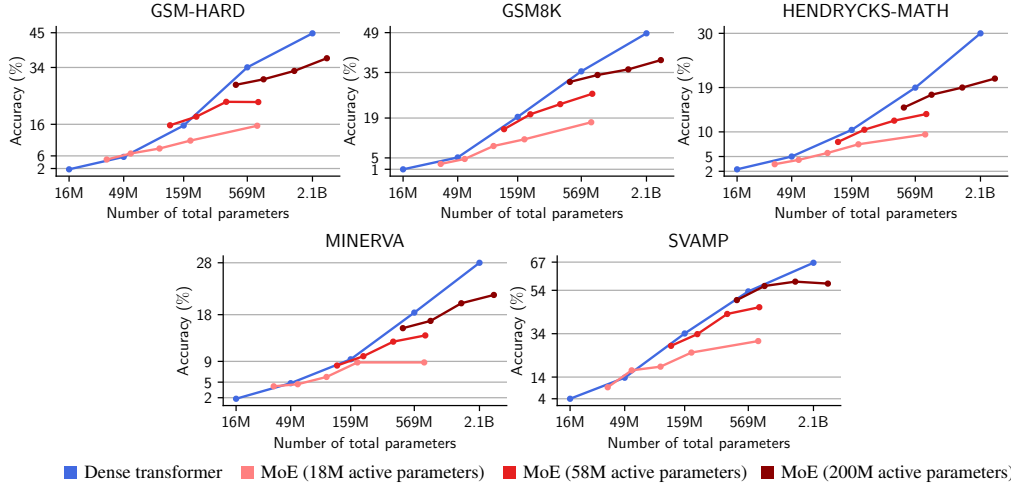


Figure 9: Downstream performance on the math benchmarks with respect to the total number of parameters of the models.

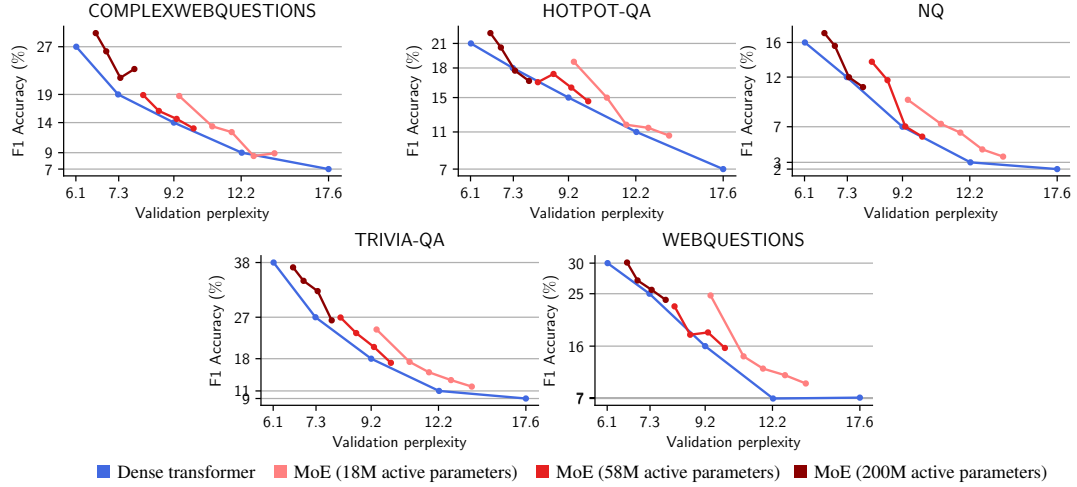


Figure 10: Downstream performance on the world knowledge tasks with respect to the validation perplexity.

## C PROOFS

### C.1 REASONING PROOFS

**Definition C.1** (Set-disjointness task). Set disjointness is the following task: given two inputs  $A, B \in \{0, 1\}^r$  for some  $r \in \mathbb{N}$ , compute  $\max_i A_i B_i$ .

Set-disjointness can be thought of as follows: Alice and Bob are given sets  $A$  and  $B$  respectively. Their objective is to determine whether they have any overlapping items in their sets.

**Lemma C.2** (Equivalence of set-disjointness and length-2 path). *The set-disjointness task is equivalent to the length-2 path task.*

*Proof.* ( $\implies$ ): Given an instance of set-disjointness, we can encode it into a length-2 path problem. Denote every item  $i$  as a vertex. Denote two extra vertices as  $A, B$ , corresponding to Alice and Bob. For every element  $i$  that Alice has, draw an edge between  $A$  and  $i$ . For every element  $i$  that Bob has, draw an edge between  $B$  to  $i$ . If and only if there are any overlapping elements, then there is

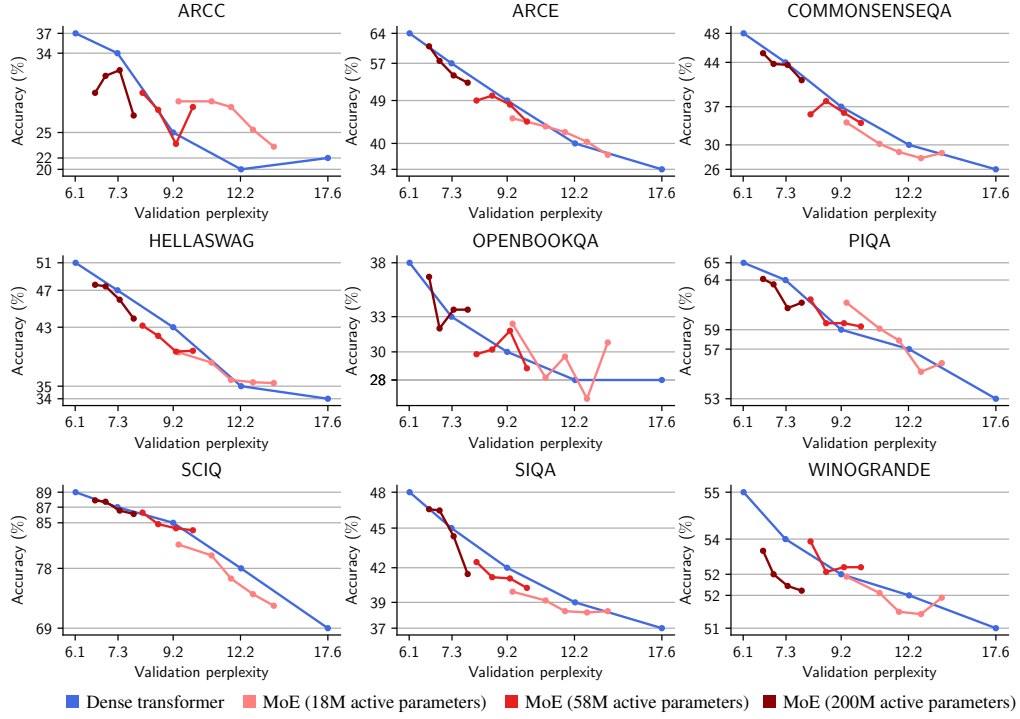


Figure 11: Performance on the commonsense tasks with respect to the validation perplexity.

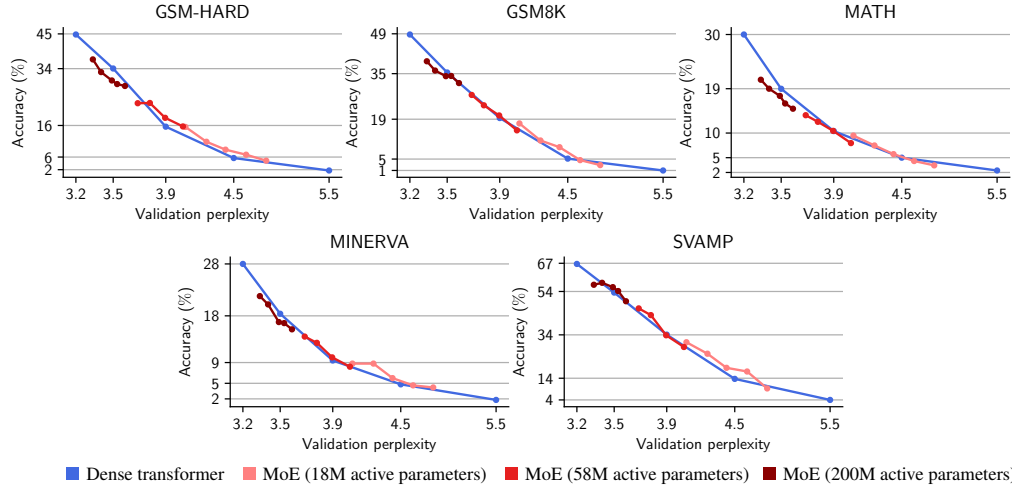


Figure 12: Downstream performance on the math benchmarks with respect to the validation perplexity.

a length-2 path from  $A$  to  $B$ . The number of elements because the number of vertices that do not belong to Alice or Bob.

( $\Leftarrow$ ): Consider an instance  $G = (V, E)$ ,  $s, d$  of length-2 path, where  $s$  is the source vertex and  $d$  is the sink vertex. For all vertices with an edge with  $s$ , put this element into Alice's set of elements. For all vertices with an edge with  $d$ , put this element into Bobs's set of elements. If and only if there is a length-2 path, then Alice and Bob's sets are overlapping. Then,  $r$  is the number of vertices.  $\square$

**Lemma C.3** (Communication complexity lower-bound on concatenated outputs). *For some sequence length, fix two disjoint subsets  $A, B \subset [N - 1]$ , and consider a single-layer transformer  $f \in \text{Transformer}_{m, H, 1}^N$  with  $O(\log N)$ -bit precision that solves set disjointness for any input  $X$  where*

$X_A$  is a function of Alice's input  $a \in \{0, 1\}^r$ ,  $X_B$  is a function of Bob's input  $b \in \{0, 1\}^r$ , and  $X_{[N] \setminus (A \cup B)}$  is fixed regardless of  $a, b$ . Then,  $f$  has width satisfying  $mH = \Omega(r / \log N)$ .

*Proof.* By re-writing the following, the remainder of the proof from [Sanford et al. \(2024\)](#) still holds.

$$\text{DISJ}(a, b) = \psi \left( [\text{softmax}(\phi(x_N)^\top Q_h K_h^\top \phi(X)) \phi(X) v_h]_{h \in [H]} \right).$$

This is because we may still use the same definition for  $Z_{h,S}, L_{h,S}$  as in the proof. Hence, this concludes the proof.  $\square$

### C.1.1 PROOF OF THEOREM 3.2

We restate the corollary.

**Theorem C.4** (Theorem 3.2). *For some input sequence  $G = (V, E)$ , fix two disjoint subsets  $A, B \subset [N - 1]$ , and consider a single-layer transformer  $f \in \text{Transformer}_{m,H,1,K}^N$  with  $O(\log N)$ -bit precision that solves length-2 path for any input  $X$  where  $X_A$  is a function of edges with the source  $s$ ,  $X_B$  is a function of edges with the destination  $d$ . Then,  $f$  has width satisfying  $mH = \Omega(|V| / \log N)$ .*

*Proof.* The proof outline is as follows:

1. Adapt Lemma 39 ([Sanford et al., 2024](#)) to support concatenation instead of addition from different attention heads.
2. The lower bound with concatenation holds for length-2 path because set-disjointness and length-2 path are equivalent.
3. Extend the result to sparse transformers.

We complete the first step with Lemma C.3. We complete the second set due to Lemma C.2. It remains to show that a router function also yields the same lower bound. We show that Lemma 39 of [Sanford et al. \(2024\)](#) can be generalized to the case in which  $\psi$  is applied according to a routing function. Specifically, consider a top-1 routing function  $r : \mathbb{R}^m \rightarrow [K]$ , and  $K$  element-wise functions  $\psi_1, \dots, \psi_K : \mathbb{R}^m \rightarrow \mathbb{R}$ . For shorthand, define:

$$Y(X_N) = [\text{softmax}(\phi(x_N)^\top Q_h K_h^\top \phi(X)) \phi(X) v_h]_{h \in [H]},$$

which is the output of the attention head prior to applying the element-wise transformation. Next, we define  $f(X_N)$  as the output when the router function  $r$  is used to select  $\psi_i$ .

$$f(X_N) = \sum_{i \in K} \mathbf{I}\{r(Y(X_N)) = i\} \psi_i(Y(X_N)).$$

Because the lower bound does not place any restrictions on the function  $\psi$  and rather argues a communication-complexity lower bound due to information from  $Y(X_N)$ , the lower bound also holds for a routing function.  $\square$

### C.1.2 PROOF OF THEOREM 3.3

We re-state Theorem 3.3 and give its proof.

**Theorem C.5** (Theorem 3.3). *For sequence length  $N$ ,  $f \in \text{Transformer}_{m,H,1}^N$  with  $O(\log N)$ -bit precision that solves length-2 path for any input  $X$ . Then, there exists a dense transformer with width  $|V|$  which solves the problem.*

*Proof.* Tokens are elements in  $\mathcal{V} = V \cup \{0\} \times V \cup \{0\}$ . The input is as follows: for vertex  $i$ , if the source shares an edge with that vertex, then the  $i$ 'th input value is  $(s, i)$ . Otherwise, it is  $(s, 0)$ . The first  $|V|$  tokens we see correspond to edges possibly shared with the source vertex. Then, the last  $|V|$  input tokens correspond to edges possibly shared with the destination vertex and share the same

format as the first  $r$  tokens. In between, we can have arbitrary edges  $(u, v)$ . We define an embedding function where  $\mathbf{e}_i$  is the  $i$ 'th standard basis vector in dimension  $r$ .

$$\begin{aligned} \phi : \mathcal{V} &\rightarrow \mathbb{R}^{|V|} \\ (u, v) &\mapsto \begin{cases} \mathbf{e}_i & \text{if } i > 0 \text{ and } u = s \text{ or } u = v \\ \mathbf{0} & \text{if } i = 0. \end{cases} \end{aligned}$$

Next, we define  $V_h \in \mathbb{R}^{|V| \times |V|}$  to be the identity matrix, and  $Q_h, V_h \in \mathbb{R}^{|V| \times |V|}$  both to have 0 everywhere. Consequently, the attention matrix is given by:

$$\left( \begin{bmatrix} 1/|V| & \dots & 1/|V| \\ \vdots & \ddots & \vdots \\ 1/|V| & & 1/|V| \end{bmatrix} \phi(X) \right)_{j,i} = \begin{cases} 2/|V| & \text{if there is a path through } ii \\ 1/|V| & \text{if one target vertex shares an edge with } i \\ 0 & \text{otherwise.} \end{cases}$$

For any entry that exceeds  $\frac{1}{|V|}$ , the correct answer is there is a length-2 path. Hence, any thresholding function which achieves this separation suffices.  $\square$

## C.2 MEMORIZATION PROOFS

In this section, we use  $d$  to denote the input dimension,  $N$  to denote the number of examples and  $n$  to denote the sequence length.

**Lemma C.6.** *Let  $x_1, x_2, \dots, x_N \in \mathbb{R}^d$  be independently sampled from the multivariate normal distribution  $\mathcal{N}(0, \sigma^2 I_d)$ , where  $\sigma > 0$  and  $I_d$  is the  $d \times d$  identity matrix. For any  $\delta \in (0, 1)$ , with probability at least  $1 - \delta$ , every pair of distinct vectors  $x_i$  and  $x_j$  satisfies*

$$|x_i^\top x_j| \leq \sigma^2 \sqrt{2d \left( 2 \ln N + \ln \frac{2}{\delta} \right)}.$$

*Proof.* We aim to bound the inner product  $x_i^\top x_j$  for each pair  $(i, j)$  with  $i \neq j$ . Since the vectors are sampled independently from  $\mathcal{N}(0, \sigma^2 I_d)$ , each component  $x_{ik}$  and  $x_{jk}$  is independently distributed as  $\mathcal{N}(0, \sigma^2)$ .

For fixed  $i \neq j$ , the inner product  $S_{ij} = x_i^\top x_j = \sum_{k=1}^d x_{ik} x_{jk}$  is the sum of  $d$  independent random variables. Each term  $x_{ik} x_{jk}$  has:

- **Mean:**

$$\mathbb{E}[x_{ik} x_{jk}] = \mathbb{E}[x_{ik}] \mathbb{E}[x_{jk}] = 0.$$

- **Variance:**

$$\text{Var}[x_{ik} x_{jk}] = \mathbb{E}[x_{ik}^2] \mathbb{E}[x_{jk}^2] = \sigma^4.$$

Since  $S_{ij}$  is a sum of independent, zero-mean random variables with variance  $\sigma^4$ , the variance of  $S_{ij}$  is:

$$\text{Var}[S_{ij}] = d\sigma^4.$$

We use the fact that  $S_{ij}$  is approximately normally distributed due to the Central Limit Theorem. For a normal distribution  $Z \sim \mathcal{N}(0, \sigma_Z^2)$ , the tail probability satisfies:

$$\mathbb{P}(|Z| \geq t) \leq 2 \exp \left( -\frac{t^2}{2\sigma_Z^2} \right).$$

Applying this to  $S_{ij}$ , we have:

$$\mathbb{P}(|x_i^\top x_j| \geq t) \leq 2 \exp \left( -\frac{t^2}{2d\sigma^4} \right).$$

There are  $\binom{N}{2} \leq \frac{N^2}{2}$  pairs of distinct vectors. Applying the union bound over all pairs:

$$\mathbb{P}(\exists i \neq j : |x_i^\top x_j| \geq t) \leq N^2 \exp\left(-\frac{t^2}{2d\sigma^4}\right).$$

To ensure that this probability is at most  $\delta$ , set:

$$N^2 \exp\left(-\frac{t^2}{2d\sigma^4}\right) \leq \delta.$$

Taking the natural logarithm:

$$-\frac{t^2}{2d\sigma^4} + 2 \ln N \leq \ln \delta.$$

Rewriting:

$$\frac{t^2}{2d\sigma^4} \geq 2 \ln N - \ln \delta.$$

Noting that  $-\ln \delta = \ln \frac{1}{\delta}$ , we have:

$$\frac{t^2}{2d\sigma^4} \geq 2 \ln N + \ln \frac{1}{\delta}.$$

Including the factor from the inequality, adjust as:

$$\frac{t^2}{2d\sigma^4} \geq 2 \ln N + \ln \frac{2}{\delta}.$$

Thus,

$$t \geq \sigma^2 \sqrt{2d \left(2 \ln N + \ln \frac{2}{\delta}\right)}.$$

Therefore, with probability at least  $1 - \delta$ , every pair of distinct vectors satisfies:

$$|x_i^\top x_j| \leq \sigma^2 \sqrt{2d \left(2 \ln N + \ln \frac{2}{\delta}\right)}.$$

□

**Lemma C.7.** Let  $x_1, x_2, \dots, x_N \in \mathbb{R}^d$  be independently sampled from the multivariate normal distribution  $\mathcal{N}(0, \sigma^2 I_d)$ . For any  $\delta \in (0, 1)$ , with probability at least  $1 - \delta$ , every vector  $x_i$  satisfies

$$\|x_i\|_\infty \leq \sigma \sqrt{2 \ln \left(\frac{2Nd}{\delta}\right)}.$$

*Proof.* Each component  $x_{ik}$  is independently distributed as  $\mathcal{N}(0, \sigma^2)$ . For a Gaussian random variable  $X \sim \mathcal{N}(0, \sigma^2)$ , the tail probability is:

$$\mathbb{P}(|X| \geq t) \leq 2 \exp\left(-\frac{t^2}{2\sigma^2}\right).$$

For a fixed vector  $x_i$ , the probability that its  $L_\infty$  norm exceeds  $t$  is:

$$\mathbb{P}(\|x_i\|_\infty \geq t) \leq 2d \exp\left(-\frac{t^2}{2\sigma^2}\right).$$



Applying the union bound over all  $N$  vectors:

$$\mathbb{P}(\exists i : \|x_i\|_\infty \geq t) \leq 2Nd \exp\left(-\frac{t^2}{2\sigma^2}\right).$$

To ensure this probability is at most  $\delta$ , set:

$$2Nd \exp\left(-\frac{t^2}{2\sigma^2}\right) \leq \delta.$$

Taking logarithms:

$$-\frac{t^2}{2\sigma^2} + \ln(2Nd) \leq \ln \delta.$$

Rewriting:

$$\frac{t^2}{2\sigma^2} \geq \ln \frac{2Nd}{\delta}.$$

Solving for  $t$ :

$$t \geq \sigma \sqrt{2 \ln \left( \frac{2Nd}{\delta} \right)}.$$

Therefore, with probability at least  $1 - \delta$ , every vector  $x_i$  satisfies:

$$\|x_i\|_\infty \leq \sigma \sqrt{2 \ln \left( \frac{2Nd}{\delta} \right)}.$$

□

**Lemma C.8.** Let  $x_1, x_2, \dots, x_N \in \mathbb{R}^d$  be independently sampled from the multivariate normal distribution  $\mathcal{N}(0, \sigma^2 I_d)$ . For any  $\delta \in (0, 1)$ , with probability at least  $1 - \delta$ , every vector  $x_i$  satisfies

$$\|x_i\|_2 \geq \sigma \sqrt{d} \left( 1 - \sqrt{\frac{2 \ln \left( \frac{N}{\delta} \right)}{d}} \right).$$

*Proof.* Each vector  $x_i$  has components that are independent  $\mathcal{N}(0, \sigma^2)$  random variables. Thus,  $\|x_i\|_2^2 = \sum_{k=1}^d x_{ik}^2$  is distributed as  $\sigma^2 \chi_d^2$ , where  $\chi_d^2$  denotes the chi-squared distribution with  $d$  degrees of freedom.

Using concentration inequalities for chi-squared distributions, for any  $\varepsilon \in (0, 1)$ :

$$\mathbb{P}(\|x_i\|_2^2 \leq \sigma^2 d(1 - \varepsilon)) \leq \exp\left(-\frac{d\varepsilon^2}{4}\right).$$

Applying the union bound over all  $N$  vectors:

$$\mathbb{P}(\exists i : \|x_i\|_2^2 \leq \sigma^2 d(1 - \varepsilon)) \leq N \exp\left(-\frac{d\varepsilon^2}{4}\right).$$

To ensure this probability is at most  $\delta$ , set:

$$N \exp\left(-\frac{d\varepsilon^2}{4}\right) \leq \delta.$$

Taking logarithms:

$$-\frac{d\varepsilon^2}{4} + \ln N \leq \ln \delta.$$

Rewriting:

$$\frac{d\varepsilon^2}{4} \geq \ln \frac{N}{\delta}.$$

Solving for  $\varepsilon$ :

$$\varepsilon \geq 2\sqrt{\frac{\ln\left(\frac{N}{\delta}\right)}{d}}.$$

Since  $\varepsilon \in (0, 1)$ , we can use the inequality  $\sqrt{1 - \varepsilon} \geq 1 - \frac{\varepsilon}{2}$  for  $\varepsilon \in (0, 1)$ . Therefore, with probability at least  $1 - \delta$ , every vector  $x_i$  satisfies:

$$\begin{aligned} \|x_i\|_2 &\geq \sigma\sqrt{d(1 - \varepsilon)} \\ &\geq \sigma\sqrt{d}\left(1 - \frac{\varepsilon}{2}\right) \\ &\geq \sigma\sqrt{d}\left(1 - \sqrt{\frac{2\ln\left(\frac{N}{\delta}\right)}{d}}\right). \end{aligned}$$

□

**Theorem C.9.** *Let  $x_1, x_2, \dots, x_N \in \mathbb{R}^d$  be independently sampled from the multivariate normal distribution  $\mathcal{N}(0, \sigma^2 I_d)$ , and let  $y_1, y_2, \dots, y_N \in \{\pm 1\}$  be arbitrary labels. Then, with probability at least  $1 - \delta$ , there exists a one-hidden-layer ReLU neural network with  $N$  neurons that correctly classifies the points  $x_i$  according to their labels  $y_i$ , i.e.,*

$$\text{sign}(f(x_i)) = y_i \quad \text{for all } i = 1, \dots, N,$$

where  $f$  is the function computed by the network. Furthermore, the  $L_\infty$ -norms of the weights and biases are bounded as follows:

- *Input weights:*  $\|w_i\|_\infty \leq \sigma\sqrt{2\ln\left(\frac{2Nd}{\delta}\right)}.$
- *Biases:*  $|b_i| \leq \sigma^2 d \left(1 + \sqrt{\frac{2\ln\left(\frac{N}{\delta}\right)}{d}}\right).$
- *Output weights:*  $|\alpha_i| = 1.$

*Proof.* We will construct a one-hidden-layer ReLU network that correctly classifies the points  $x_i$  with the specified labels  $y_i$ . The network has the following structure:

- **Hidden layer:** Consists of  $N$  neurons with weights  $w_i \in \mathbb{R}^d$  and biases  $b_i$ .
- **Output layer:** Computes the function  $f(x) = \sum_{i=1}^N \alpha_i \text{ReLU}(w_i^\top x + b_i)$ , where  $\alpha_i = y_i$ .

### Step 1: High-Probability Bounds

From Lemmas C.6, C.7, and C.8, with probability at least  $1 - \delta$ , the following hold simultaneously for all  $i \neq j$ :

1. **Bound on  $\|x_i\|_\infty$**  (Lemma C.7):

$$\|x_i\|_\infty \leq \sigma \sqrt{2 \ln \left( \frac{2Nd}{\delta} \right)}.$$

2. **Lower bound on  $\|x_i\|_2$**  (Lemma C.8):

$$\|x_i\|_2 \geq \sigma \sqrt{d} \left( 1 - \sqrt{\frac{2 \ln \left( \frac{N}{\delta} \right)}{d}} \right).$$

3. **Bound on  $|x_i^\top x_j|$**  (Lemma C.6):

$$|x_i^\top x_j| \leq \sigma^2 \sqrt{2d \left( 2 \ln N + \ln \frac{2}{\delta} \right)}.$$

### Step 2: Constructing the Network

We define the weights and biases as follows:

- **Input weights:**  $w_i = x_i$ .
- **Biases:**  $b_i = -x_i^\top x_i + s$ , where  $s = \frac{\sigma^2 d}{2}$ .
- **Output weights:**  $\alpha_i = y_i$ .

### Step 3: Network Output on Training Points

For each training point  $x_j$ , the pre-activation of the  $i$ -th hidden neuron is:

$$z_{ij} = w_i^\top x_j + b_i = x_i^\top x_j - x_i^\top x_i + s.$$

We consider two cases:

**Case 1:**  $i = j$

$$z_{jj} = x_j^\top x_j - x_j^\top x_j + s = s > 0.$$

Therefore,

$$\text{ReLU}(z_{jj}) = s.$$

**Case 2:**  $i \neq j$

Using the bounds from Step 1:

$$\begin{aligned} z_{ij} &= x_i^\top x_j - x_i^\top x_i + s \\ &\leq |x_i^\top x_j| - \|x_i\|_2^2 + s \\ &\leq \sigma^2 \sqrt{2d \left( 2 \ln N + \ln \frac{2}{\delta} \right)} - \sigma^2 d \left( 1 - \sqrt{\frac{2 \ln \left( \frac{N}{\delta} \right)}{d}} \right)^2 + s. \end{aligned}$$

Simplify the expression (assuming  $d$  is large enough that terms involving  $\frac{\ln N}{d}$  are small):

$$\text{Let } \varepsilon = \sqrt{\frac{2 \ln \left( \frac{N}{\delta} \right)}{d}}, \text{ and } \gamma = \sqrt{\frac{2 \left( 2 \ln N + \ln \frac{2}{\delta} \right)}{d}}.$$

Then:

$$z_{ij} \leq \sigma^2 d \left( \gamma - (1 - \varepsilon)^2 \right) + s.$$

Note that:

$$(1 - \varepsilon)^2 = 1 - 2\varepsilon + \varepsilon^2.$$

Therefore,

$$z_{ij} \leq \sigma^2 d \left( \gamma - 1 + 2\varepsilon - \varepsilon^2 \right) + s.$$

Assuming  $\varepsilon$  and  $\varepsilon^2$  are small, and  $\gamma$  is small compared to 1 (since  $d$  is large), we have:

$$z_{ij} \leq -c\sigma^2 d,$$

for some positive constant  $c > 0$ . Therefore,

$$\text{ReLU}(z_{ij}) = 0.$$

#### Step 4: Final Output

The network output for  $x_j$  is:

$$f(x_j) = \sum_{i=1}^N \alpha_i \text{ReLU}(z_{ij}) = y_j s + \sum_{i \neq j} y_i \cdot 0 = y_j s.$$

Since  $s > 0$ , the sign of  $f(x_j)$  matches  $y_j$ :

$$\text{sign}(f(x_j)) = \text{sign}(y_j s) = y_j.$$

#### Step 5: Bounding the Weights and Biases

- **Input Weights:** From Lemma C.7:

$$\|w_i\|_\infty = \|x_i\|_\infty \leq \sigma \sqrt{2 \ln \left( \frac{2Nd}{\delta} \right)}.$$

- **Biases:** Using the bound on  $\|x_i\|_2$  from Lemma C.8:

$$\begin{aligned} |b_i| &= |-x_i^\top x_i + s| \\ &\leq \|x_i\|_2^2 + s \\ &\leq \left( \sigma \sqrt{d} (1 - \varepsilon) \right)^2 + \frac{\sigma^2 d}{2} \\ &= \sigma^2 d \left( (1 - \varepsilon)^2 + \frac{1}{2} \right) \\ &= \sigma^2 d \left( 1 - 2\varepsilon + \varepsilon^2 + \frac{1}{2} \right) \\ &\leq \sigma^2 d \left( \frac{3}{2} - 2\varepsilon \right). \end{aligned}$$

- **Output Weights:**  $|\alpha_i| = |y_i| = 1$ .

□

**Theorem C.10.** Let  $X_1, X_2, \dots, X_N \in \mathbb{R}^{n \times d}$  be  $N$  sequences of length  $n$ , where each token  $X_{ik} \in \mathbb{R}^d$  is independently sampled from the multivariate normal distribution  $\mathcal{N}(0, I_d)$ . Let  $y_1, y_2, \dots, y_N \in \{\pm 1\}$  be arbitrary labels. Then, with probability at least  $1 - \delta$ , there exists a one-layer transformer with inner dimension  $N$  *Clara: should probably use a different variable* that, when applied to each sequence  $X_i$ , outputs at the last token a value whose sign matches  $y_i$ , i.e.,

$$\text{sign}(f(X_i)) = y_i \quad \text{for all } i = 1, \dots, N,$$

where  $f$  is the function computed by the transformer. Furthermore, the  $L_\infty$ -norms of the weights and biases of the transformer are explicitly bounded as follows:

- The  $L_\infty$ -norm of all weights in the attention mechanism is at most 1.
- The  $L_\infty$ -norm of the feed-forward weights is at most

$$\|W_{ff}\|_\infty \leq \frac{1}{\sqrt{n}} \sqrt{2 \ln \left( \frac{2Nd}{\delta} \right)}.$$

- The  $L_\infty$ -norm of the feed-forward biases is at most

$$\|b_{ff}\|_\infty \leq \frac{d}{n} \left( 1 + \sqrt{\frac{2 \ln \left( \frac{N}{\delta} \right)}{d}} \right).$$

- The output weights satisfy  $|\alpha_i| = 1$  for all  $i$ .

*Proof.* We will construct a one-layer transformer with inner dimension  $N$  that correctly classifies the sequences  $X_i$  according to their labels  $y_i$ . The transformer consists of:

- **Self-Attention Layer:** Configured to compute the average of the input tokens at the last position.
- **Feed-Forward Network:** Applied at the last token to classify the averaged input.

### Step 1: Configure Self-Attention to Compute Token Averages

Our goal is to compute the average of the input tokens  $X_{i1}, X_{i2}, \dots, X_{in}$  at the last token position. To achieve uniform attention, we set the query and key matrices to zero:

- $W^Q = 0 \in \mathbb{R}^{d \times d_k}$
- $W^K = 0 \in \mathbb{R}^{d \times d_k}$

Since  $Q_t = W^Q X_{it} = 0$  and  $K_{t'} = W^K X_{it'} = 0$  for all tokens  $t, t'$ , the attention scores become:

$$\text{AttentionScore}_{t,t'} = \frac{Q_t^\top K_{t'}}{\sqrt{d_k}} = 0.$$

The softmax of a vector of zeros yields uniform attention weights:

$$\alpha_{t,t'} = \frac{1}{n}.$$

We set the value matrix  $W^V = I_d$  (the identity matrix), so the output of the attention layer at the last token  $t = n$  is:

$$h_n = \sum_{t'=1}^n \alpha_{n,t'} V_{t'} = \frac{1}{n} \sum_{t'=1}^n X_{it'} = S_i,$$



where  $S_i \in \mathbb{R}^d$  is the average of the input tokens for sequence  $X_i$ :

$$S_i = \frac{1}{n} \sum_{k=1}^n X_{ik}.$$

### Step 2: Distribution of $S_i$

Since each  $X_{ik}$  is independently sampled from  $\mathcal{N}(0, I_d)$ , the average  $S_i$  is distributed as:

$$S_i \sim \mathcal{N}\left(0, \frac{1}{n} I_d\right).$$

### Step 3: Apply the Feed-Forward Network Theorem

We now apply the previous theorem (Theorem C.9) to the vectors  $S_i$ . Specifically, since  $S_i$  are independently sampled from  $\mathcal{N}\left(0, \frac{1}{n} I_d\right)$ , we set  $\sigma = \frac{1}{\sqrt{n}}$  in Theorem C.9. The theorem guarantees that, with probability at least  $1 - \delta$ , there exists a one-hidden-layer ReLU neural network with  $N$  neurons that correctly classifies the vectors  $S_i$  according to their labels  $y_i$ , i.e.,

$$\text{sign}(f(S_i)) = y_i \quad \text{for all } i = 1, \dots, N,$$

where

$$f(S) = \sum_{i=1}^N \alpha_i \text{ReLU}(w_i^\top S + b_i),$$

with  $\alpha_i = y_i$ .

### Step 4: Bounding the Weights and Biases

From Theorem C.9, with  $\sigma = \frac{1}{\sqrt{n}}$ , the  $L_\infty$ -norms of the weights and biases are bounded as follows:

- **Input Weights:**

$$\|w_i\|_\infty \leq \frac{1}{\sqrt{n}} \sqrt{2 \ln \left( \frac{2Nd}{\delta} \right)}.$$

- **Biases:**

$$|b_i| \leq \frac{1}{n} d \left( 1 + \sqrt{\frac{2 \ln \left( \frac{N}{\delta} \right)}{d}} \right).$$

- **Output Weights:**  $|\alpha_i| = |y_i| = 1$ .

### Step 5: Mapping to Transformer Architecture

We design the feed-forward network at the last token to simulate the ReLU network operating on  $S_i$ :

- **Feed-Forward Network at Last Token:** Consists of weights  $W_{\text{ff}} \in \mathbb{R}^{d \times N}$  and biases  $b_{\text{ff}} \in \mathbb{R}^N$ , where the  $i$ -th column of  $W_{\text{ff}}$  is  $w_i$ , and the  $i$ -th element of  $b_{\text{ff}}$  is  $b_i$ .
- **Output Layer:** Computes  $f(X_i) = \alpha^\top \text{ReLU}(W_{\text{ff}}^\top S_i + b_{\text{ff}})$ , where  $\alpha_i = y_i$ .

### Step 6: Bounding the Transformer Weights

The  $L_\infty$ -norms of the transformer weights and biases are explicitly bounded:

- **Attention Weights:** Since  $W^Q = 0$  and  $W^K = 0$ , their  $L_\infty$ -norms are zero. The value matrix  $W^V = I_d$  has  $L_\infty$ -norm equal to 1.

- **Feed-Forward Weights:**

$$\|W_{\text{ff}}\|_{\infty} = \max_{i,k} |w_{ik}| \leq \frac{1}{\sqrt{n}} \sqrt{2 \ln \left( \frac{2Nd}{\delta} \right)}.$$

- **Feed-Forward Biases:**

$$\|b_{\text{ff}}\|_{\infty} = \max_i |b_i| \leq \frac{d}{n} \left( 1 + \sqrt{\frac{2 \ln \left( \frac{N}{\delta} \right)}{d}} \right).$$

- **Output Weights:**  $|\alpha_i| = 1$ .

### Step 7: Network Output on Sequences

For each sequence  $X_i$ , the transformer computes:

1. **Attention Layer:** Outputs  $S_i$  at the last token.
2. **Feed-Forward Network:** Computes

$$h_i = \text{ReLU}(W_{\text{ff}}^{\top} S_i + b_{\text{ff}}) \in \mathbb{R}^N.$$

3. **Final Output:**

$$f(X_i) = \alpha^{\top} h_i = \sum_{j=1}^N y_j \text{ReLU}(w_j^{\top} S_i + b_j).$$

Since the feed-forward network at the last token simulates the ReLU network from Step 3, we have:

$$\text{sign}(f(X_i)) = \text{sign}(f(S_i)) = y_i.$$

### Conclusion

With the constructed transformer, all sequences  $X_i$  are correctly classified according to their labels  $y_i$ , and the  $L_{\infty}$ -norms of the weights and biases are explicitly bounded as specified.

□

**Theorem C.11.** *Let  $X_1, X_2, \dots, X_N \in \mathbb{R}^{n \times d}$  be  $N$  sequences of length  $n$ , where each token  $X_{ik} \in \mathbb{R}^d$  is independently sampled from the multivariate normal distribution  $\mathcal{N}(0, I_d)$ . Let  $y_1, y_2, \dots, y_N \in \{\pm 1\}$  be arbitrary labels. Then, with probability at least  $1 - \delta$ , there exists a one-layer Mixture-of-Experts (MoE) transformer with  $K$  experts, each having  $O\left(\frac{N}{K}\right)$  neurons, that, when applied to each sequence  $X_i$ , outputs at the last token a value whose sign matches  $y_i$ , i.e.,*

$$\text{sign}(f(X_i)) = y_i \quad \text{for all } i = 1, \dots, N.$$

*Furthermore, the  $L_{\infty}$ -norms of the weights and biases of the transformer are explicitly bounded, and the bit-complexity (number of bits per parameter) is*

$$O\left(\log(nd) + \log \ln \left( \frac{NK}{\delta} \right)\right).$$

*Proof.* We construct a one-layer MoE transformer with  $K$  experts to classify the sequences  $X_i$  according to their labels  $y_i$ . The transformer operates as follows:

1. **Self-Attention Layer:** Configured to compute the average of the input tokens at the last position.

2. **Routing Function:** Assigns each sequence to one of the  $K$  experts based on a routing decision.
3. **Expert Networks:** Each expert processes its assigned sequences using a feed-forward network.

### Step 1: Configure Self-Attention to Compute Token Averages

As in the previous theorem, we set the query and key matrices to zero to achieve uniform attention weights:

- $W^Q = 0 \in \mathbb{R}^{d \times d_k}$
- $W^K = 0 \in \mathbb{R}^{d \times d_k}$

The output at the last token  $t = n$  is the average of the input tokens:

$$h_n = \frac{1}{n} \sum_{k=1}^n X_{ik} = S_i,$$

where  $S_i \sim \mathcal{N}\left(0, \frac{1}{n} I_d\right)$ .

### Step 2: Define Routing Vectors and Assign Inputs to Experts

We define routing vectors  $r_1, r_2, \dots, r_K \in \mathbb{R}^d$ , where each  $r_j$  is independently sampled from  $\mathcal{N}(0, I_d)$ . For each sequence  $X_i$ , we compute routing scores:

$$s_{ij} = r_j^\top S_i, \quad \text{for } j = 1, \dots, K.$$

The sequence  $X_i$  is assigned to expert  $j^*$  where:

$$j^* = \arg \max_{1 \leq j \leq K} s_{ij}.$$

Since  $S_i \sim \mathcal{N}\left(0, \frac{1}{n} I_d\right)$  and  $r_j \sim \mathcal{N}(0, I_d)$ , the routing scores  $s_{ij}$  are independent and distributed as  $\mathcal{N}\left(0, \frac{1}{n}\right)$ .

### Step 3: Balance Inputs Among Experts

For each input  $S_i$ , the probability that it is assigned to expert  $j$  is:

$$\mathbb{P}(X_i \text{ assigned to expert } j) = \frac{1}{K}.$$

Let  $N_j$  denote the number of inputs assigned to expert  $j$ . Since assignments are independent,  $N_j$  follows a binomial distribution  $\text{Binomial}(N, \frac{1}{K})$ .

Using Hoeffding's inequality, for any  $\varepsilon > 0$ :

$$\mathbb{P}\left(\left|N_j - \frac{N}{K}\right| \geq \varepsilon N\right) \leq 2 \exp(-2\varepsilon^2 N).$$

Set  $\varepsilon = \sqrt{\frac{\ln(2K/\delta)}{2N}}$ . Then,

$$\mathbb{P}\left(\left|N_j - \frac{N}{K}\right| \geq \varepsilon N\right) \leq \frac{\delta}{K}.$$

Applying the union bound over all experts:

$$\mathbb{P}\left(\exists j : \left|N_j - \frac{N}{K}\right| \geq \varepsilon N\right) \leq \delta.$$

Therefore, with probability at least  $1 - \delta$ , each expert receives at most

$$N_j \leq \frac{N}{K} + \varepsilon N = \frac{N}{K} + N \sqrt{\frac{\ln(2K/\delta)}{2N}} = \frac{N}{K} + \sqrt{\frac{N \ln(2K/\delta)}{2}}.$$

Since  $N$  is large,  $N_j = O\left(\frac{N}{K}\right)$ .

#### Step 4: Apply the Feed-Forward Network Theorem to Each Expert

Within each expert  $j$ , we have  $N_j$  inputs  $S_i$  assigned to it. We apply Theorem C.9 (from the previous result) to construct a feed-forward ReLU network that correctly classifies these inputs. Specifically:

- Inputs: The vectors  $S_i$  assigned to expert  $j$ , each sampled from  $\mathcal{N}\left(0, \frac{1}{n} I_d\right)$ .
- Labels: The corresponding  $y_i$  for these inputs.
- Network Size: The network uses  $N_j$  neurons.

From Theorem C.9 (with  $\sigma = \frac{1}{\sqrt{n}}$  and  $N$  replaced by  $N_j$ ), with probability at least  $1 - \frac{\delta}{K}$ , the network correctly classifies all inputs assigned to expert  $j$ . Applying the union bound over all experts, with probability at least  $1 - \delta$ , all experts correctly classify their assigned inputs.

#### Step 5: Bounding the Weights and Biases

From Theorem C.9, the  $L_\infty$ -norms of the weights and biases in each expert are bounded:

- **Input Weights:**

$$\|w_i\|_\infty \leq \frac{1}{\sqrt{n}} \sqrt{2 \ln \left( \frac{2N_j d}{\delta/K} \right)} \leq \frac{1}{\sqrt{n}} \sqrt{2 \ln \left( \frac{2NdK}{\delta} \right)}.$$

- **Biases:**

$$|b_i| \leq \frac{d}{n} \left( 1 + \sqrt{\frac{2 \ln \left( \frac{N_j}{\delta/K} \right)}{d}} \right) \leq \frac{d}{n} \left( 1 + \sqrt{\frac{2 \ln \left( \frac{NK}{\delta} \right)}{d}} \right).$$

- **Output Weights:**  $|\alpha_i| = 1$ .

#### Step 6: Bounding the Bit-Complexity

To determine the bit-complexity per parameter, we need to calculate the number of bits required to represent the weights and biases with sufficient precision.

Let  $\epsilon$  be the desired precision for representing each parameter.

##### Weights:

The maximum absolute value of the weights is:

$$M_w = \frac{1}{\sqrt{n}} \sqrt{2 \ln \left( \frac{2NdK}{\delta} \right)}.$$

The number of bits required per weight parameter is:

$$\begin{aligned}
\text{Bits}_w &= O\left(\log\left(\frac{M_w}{\epsilon}\right)\right) \\
&= O\left(\log\left(\frac{1}{\sqrt{n}}\sqrt{2\ln\left(\frac{2NdK}{\delta}\right)}\frac{1}{\epsilon}\right)\right) \\
&= O\left(\log\left(\frac{1}{\sqrt{n}}\right) + \frac{1}{2}\log\left(2\ln\left(\frac{2NdK}{\delta}\right)\right) + \log\left(\frac{1}{\epsilon}\right)\right) \\
&= O\left(\left(-\frac{1}{2}\log n\right) + \frac{1}{2}\log\ln\left(\frac{NK}{\delta}\right) + \frac{1}{2}\log(2\ln d) + \log\left(\frac{1}{\epsilon}\right)\right).
\end{aligned}$$

Simplifying, we have:

$$\text{Bits}_w = O\left(\log n + \log d + \log\ln\left(\frac{NK}{\delta}\right) + \log\left(\frac{1}{\epsilon}\right)\right).$$

Note that the negative term  $-\frac{1}{2}\log n$  becomes negligible in the overall  $O$  notation, as we are concerned with the total number of bits required.

#### Biases:

The maximum absolute value of the biases is:

$$M_b = \frac{d}{n} \left(1 + \sqrt{\frac{2\ln\left(\frac{NK}{\delta}\right)}{d}}\right) \leq \frac{d}{n} \left(1 + \sqrt{\frac{2\ln\left(\frac{NK}{\delta}\right)}{d}}\right).$$

Since  $\sqrt{\frac{2\ln\left(\frac{NK}{\delta}\right)}{d}}$  is small for large  $d$ , we can approximate  $M_b \approx \frac{d}{n}$ . The number of bits required per bias parameter is:

$$\begin{aligned}
\text{Bits}_b &= O\left(\log\left(\frac{M_b}{\epsilon}\right)\right) \\
&= O\left(\log\left(\frac{d}{n\epsilon}\right)\right) \\
&= O\left(\log d + \log n + \log\left(\frac{1}{\epsilon}\right)\right).
\end{aligned}$$

#### Total Bit-Complexity per Parameter:

Combining the bits required for weights and biases, the bit-complexity per parameter is:

$$\text{Bits} = O\left(\log n + \log d + \log\ln\left(\frac{NK}{\delta}\right) + \log\left(\frac{1}{\epsilon}\right)\right).$$

Since  $\epsilon$  is a constant precision (e.g., machine epsilon), we can omit  $\log\left(\frac{1}{\epsilon}\right)$  in the  $O$  notation.

Therefore, the bit-complexity per parameter depends logarithmically on  $n$  and  $d$ , and logarithmically on the logarithm of  $N$ ,  $K$ , and  $1/\delta$ . This means that  $n$  and  $d$  are inside a single logarithm, while  $N$ ,  $K$ , and  $1/\delta$  are inside a double logarithm.

#### Step 7: Final Transformer Architecture

The MoE transformer consists of:

- **Attention Layer:** Computes  $S_i = \frac{1}{n} \sum_{k=1}^n X_{ik}$  at the last token.

- **Routing Function:** Assigns  $S_i$  to expert  $j^* = \arg \max_j r_j^\top S_i$ .
- **Experts:** Each expert  $j$  has its own feed-forward network with weights and biases as constructed in Step 4.
- **Output:** For each  $X_i$ , the transformer outputs  $f(X_i) = f_j(S_i)$  where  $f_j$  is the function computed by expert  $j$ .

### Conclusion

With the constructed MoE transformer, all sequences  $X_i$  are correctly classified according to their labels  $y_i$ . The total number of neurons across all experts is:

$$\sum_{j=1}^K N_j = N,$$

since each input is assigned to exactly one expert. The  $L_\infty$ -norms of the weights and biases are explicitly bounded, and the bit-complexity per parameter is

$$O\left(\log(nd) + \log \ln\left(\frac{NK}{\delta}\right)\right).$$

This completes the proof. □

*Proof of Theorem 3.6.* Let  $c$  be the number of bits used for encoding each parameters (and we assume that  $c$  is logarithmic in the problem parameters). Denote by  $\mathcal{H}$  the class of all transformers with  $W$  parameters and  $c$  bits per parameters. Since  $\mathcal{H}$  is a finite class, where each function in the class can be encoded with  $cW$  bits, we have  $|\mathcal{H}| \leq 2^{cW}$ . Let  $X^1, \dots, X^N \in \mathbb{R}^{n \times d}$  be the  $N$  input points. Assume a  $\mathcal{H}$  can solve the memorization task. Then, for every choice of  $y_1, \dots, y_N \in \{\pm 1\}$ , there exists a transformer  $f \in \mathcal{H}$  s.t.  $f(X_i) = y_i$  for all  $i \in [N]$ . There are  $2^N$  possible assignments for  $y_1, \dots, y_N$  and therefore there are at least  $2^N$  different functions in  $\mathcal{H}$ . So, we get  $2^N \leq |\mathcal{H}| \leq 2^{cW}$  and therefore  $W \geq N/c$ . □